Hardware-Accelerated Simulated Radiography

Daniel Laney*
Lawrence Livermore National Laboratory

Steven P. Callahan †
University of Utah

Nelson Max ‡
Lawrence Livermore National Laboratory

Cláudio T. Silva §
University of Utah

Steven Langer
Lawrence Livermore National Laboratory

Randall Frank
Computational Engineering International

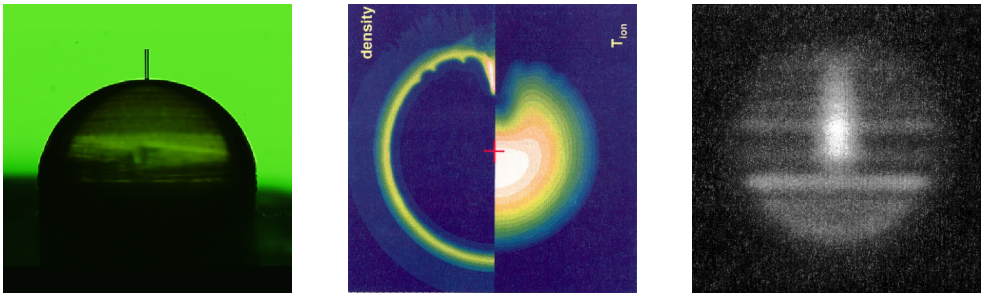


Figure 1: (a) A fuel capsule for inertial confinement fusion showing the fill tube by which Hydrogen fuel is injected. (b) Simulations and experiments indicate that during capsule compression the fill tube may cause a jet to form that may lead to reduced yield. (c) Simulated radiographs are used to design diagnostics that can detect the jet during experiments and lead to quantitative measurements of position and velocity.

Abstract

We present the application of hardware accelerated volume rendering algorithms to the simulation of radiographs as an aid to scientists designing experiments, validating simulation codes, and understanding experimental data. The techniques presented take advantage of 32-bit floating point texture capabilities to obtain solutions to the radiative transport equation for X-rays. The hardware accelerated solutions are accurate enough to enable scientists to explore the experimental design space with greater efficiency than the methods currently in use. An unsorted hexahedron projection algorithm is presented for curvilinear hexahedral meshes that produces simulated radiographs in the absorption-only regime. A sorted tetrahedral projection algorithm is presented that simulates radiographs of emissive materials. We apply the tetrahedral projection algorithm to the simulation of experimental diagnostics for inertial confinement fusion experiments on a laser at the University of Rochester.

CR Categories: I.3.5 [Computer Graphics];

Keywords: volume rendering, hardware acceleration

1 Introduction

In this paper we present modifications to hardware accelerated volume rendering algorithms that enable the quantitative simulation of radiographs. Hardware accelerated algorithms cannot match the accuracy of a software solution utilizing double precision arithmetic. Our goal is to approach the accuracy of such techniques while reducing the time to solution, thereby enabling new methods of interacting with experimental and simulated radiographs.

Radiography is a diagnostic technique used in industrial and scientific applications. In the industrial setting, high-energy X-rays are passed through objects as in a medical X-ray. In these cases the attenuation of the X-rays dominates. In high-energy physics applications radiographs may include both attenuation and emission of X-rays by the object being imaged. In both cases it is necessary to adjust the parameters of the radiographic set-up to optimize the resulting image. Depending on the application, this optimization process may be extensive. Designing an experimental diagnostic may require a simulation of the experiment, followed by iteratively modifying the diagnostic parameters in order to determine a set of parameters that captures the features of the experiment.

Fast simulation of radiographs can also aid in understanding radiographical images obtained experimentally. The positions and shapes of interesting features in particular radiographs may not be easy to identify, or their causes may be hard to determine due to the lack of depth information in the experimentally obtained image. However, if a simulation of the experiment is available, simulated radiographs can be generated with varying orientations and material properties. Such techniques may also be useful for tomography applications.

The addition of 32-bit IEEE 754 floating point capability to commodity graphics hardware has opened up the possibility of using this hardware for simulation. GPU-based linear algebra [10] and computational fluid dynamics [11] are two examples in an increasing set of algorithms that have been mapped to graphics hardware. At Lawrence Livermore National Laboratory these new techniques have not been adopted by computational scientists mainly due to the difficulty of programming graphics hardware and the absence of 64-bit floating point arithmetic. We have found that 32-bit floating point arithmetic is sufficient for the solution of the radiation transport equation for X-rays although there is a reduction in accuracy. The use of 16-bit floating point frame buffers is an option for less

*e-mail: dlaney@llnl.gov
†e-mail: stevec@sci.utah.edu
‡e-mail: max2@llnl.gov
§e-mail: csilva@cs.utah.edu

accurate but higher performance implementations. The opacities and emissivities encountered in some domains can span six orders of magnitude or more, causing problems when compositing many small values. Using 32-bit arithmetic appears to mitigate these errors. Solving the transport equation allows us to create simulated radiographs of proposed experiment diagnostics, and also to accelerate the simulation of industrial radiographs.

We describe an application involving experiment design for inertial confinement fusion for which faster response times can enhance the ability of scientists to design experimental diagnostics. The accelerated techniques enable faster turn-around for a physicist exploring the design space of a particular radiographic apparatus.

2 Previous Work

Simulated radiography tools for CAD models have been developed to enable efficient use of non-destructive evaluation for engineering applications [6, 20]. Koenig [8] used software raytracing techniques on CAD geometry to study the 3-D acquisition of defects imaged in experimental radiographs. Software raytracing techniques have been used in simulations of radiograph equipment [5]. Bonin *et al.* [1] developed a radiographical simulation based on Monte Carlo techniques. In many cases these systems are used for defect detection, in which many parameters must be adjusted in order to determine the defect sizes that can be detected with a given radiographic system. Such a use-case in not well suited to the present work, in which accuracy is traded off for simulation time to enable the detection of large features in physics experiments.

Volume rendering unstructured grids has been the subject of much research in the visualization community. Due to recent advances in programmable graphics hardware, techniques have been developed to shift much of the computation to the GPU for better performance. The Projected Tetrahedra algorithm [16] was the first to show how to render tetrahedral cells using the traditional 3D polygon-rendering pipeline. Roettger and Ertl [15] show the efficiency of the GPU for compositing ray integrals of arbitrary unstructured polyhedra using an emissive optical model. Weiler *et al.* [17, 18] demonstrate a ray-caster implemented entirely on the GPU with significant improvement in performance over software algorithms. The ray-caster traverses the tetrahedral mesh by storing the topology in textures which are kept in GPU memory. A more recent technique by Callahan *et al.* [2] removes the necessity for topological structure by volume rendering the collection of triangles that compose the mesh. Their algorithm works in both object- and image-space by approximately sorting the geometry on the CPU, then finalizing the sort and compositing the fragments on the GPU. This provides a fast, efficient, and simple solution to volume rendering unstructured grids. We extend on the work of Callahan *et al.* and show that our work can be mapped to hardware-assisted volume rendering algorithms.

3 Radiography

The radiative transfer equation for mono-energetic photons of frequency ν is

$$\frac{\mathrm{d}I_\nu}{\mathrm{d}s} = J_\nu - K_\nu I_\nu \tag{1}$$

where I_ν is the specific intensity (or radiance, or spectral radiance) in units of power per unit area per steradian per frequency, s is the position, K_ν is the extinction coefficient (or opacity) with units of inverse length, and J_ν is the specific emissivity in units of power per unit volume per unit solid angle per frequency.

Introducing the differential optical thickness $\mathrm{d}\tau_\nu = K_\nu \mathrm{d}s$ gives:

$$\frac{\mathrm{d}I_\nu}{\mathrm{d}\tau_\nu} = S_\nu - I_\nu \tag{2}$$

where $S_\nu = J_\nu / K_\nu$ is the source function with units of specific intensity. In a region with constant K_ν and J_ν the solution is:

$$I_\nu = I_{\nu 0} e^{-\tau} + (1 - e^{-\tau}) S_\nu \tag{3}$$

where $I_{\nu 0}$ is the specific intensity entering the region and the accumulated attenuation, or optical thickness τ (frequency subscript elided) is given by:

$$\tau = \int K_\nu \mathrm{d}s \tag{4}$$

Given a mesh in which the values of K_ν and J_ν are constant over each cell, I_ν can be integrated along a linear path through the mesh by solving the following first order recurrence:

$$I_\nu[i+1] = e^{-\tau[i]} I_\nu[i] + (1 - e^{-\tau[i]}) S_\nu[i] \tag{5}$$

where i indexes each cell in order from farthest to nearest with respect to the detector, and in this context J_ν, K_ν, and τ are understood as predefined per-cell quantities indexed by i.

In many cases a multi-group approximation is used to capture effects of photons of different energies. In the present work, the frequency specific quantities I_ν, J_ν, K_ν, and τ_ν are replaced by piece-wise constant approximations over a set of energy bins. The recurrence in equation 5 is then solved for each energy group using the piece-wise constant values.

4 Absorption-Only Radiographs

In industrial and medical settings, the self emission of X-rays from the subject can be neglected. The subject is back-lit by an X-ray source and an image is obtained either on film or detector. Setting the emissivity J_ν to zero in equation 5 produces

$$I_\nu[i+1] = e^{-\tau[i]} I_\nu[i] \tag{6}$$

Solving this recurrence results in a simple product of the exponential terms, which can be found from a sum of the optical thicknesses of all cells projecting onto detector pixels. The optical thickness can be computed in hardware, using the polyhedron projection method, by summing the attenuation effects of each cell. Since addition is commutative, the global back to front visibility sort usually required for the polyhedron projection method is not necessary. For a tetrahedral mesh, we can use the method of Shirley and Tuchman [16] to divide the image plane projection of each tetrahedron into triangle fans for hardware rendering. One can divide any polyhedral mesh into tetrahedra in a consistent way using the method of Max et al. [13]. However, many of our meshes are curvilinear, and there is a large saving in mesh storage and rendering time if the hexahedral cells can be projected directly. We use the method of Max [12], which describes how to define triangle fans or strips for all projections of a non-degenerate hexahedron, as long as the projection of each face is a convex quadrilateral. Hexahedral cells which are degenerate because they have coinciding vertices, or which have faces whose projections are non-convex, are divided up into tetrahedra. Note, if these tetrahedra share non-planar faces with another cell which is not subdivided, gaps or overlaps in the viewing ray segments can occur, introducing small errors in the total absorption.

Our users require floating point accuracy in the attenuation computation. Unfortunately, although current graphics chips can do internal fragment computations with 32-bit floating point accuracy, the compositing operations to the framebuffer, which are necessary to add up the attenuation effects of all the cells, are not possible with this accuracy. Therefore, the attenuation for each cell was added in a simple fragment program (see [9]). The attenuation for each vertex was the extinction coefficient K_ν times the thickness,

which was computed in software, using the geometry of the cell. The accumulated attenuation was taken from an image-aligned texture, and the attenuation for the current cell was interpolated by the hardware from the vertex data. The floating point sum was written to a floating point off-screen P-buffer, which was copied to the texture for input to the next iteration. This copying is necessary to keep the texture current, because writes to any output buffer, including a texture, are not guaranteed to be in polygon order, and also because the texture caching mechanism does not account for textures changing on the fly. We can accumulate the attenuation for four different X-ray wavelength bands in parallel, using the red, green, blue, and alpha channels.

Copies from the P-buffer to a texture are slow, so it is impractical to do the copy after rendering every cell. Therefore we attempted to render as many non-overlapping cells to the P-buffer as possible, in order to minimize the copying. This was done in multiple passes. In the first pass, all the hexahedra were tested to see if they could be projected as a whole, as described above, and if not, they were subdivided into five tetrahedra. A logically circular list of pointers to all cells was formed, and for each, an image plane bounding box was saved. Then in subsequent passes of writing the P-buffer, this list was traversed in jumps of a size relatively prime to the list's length, so that every cell would be touched once before any was repeated, and yet adjacent cells would not be touched consecutively, in order to reduce the chance of overlap. Cells whose bounding boxes did not overlap any others rendered in the current pass could be rendered into the P-buffer.

Overlapping bounding boxes were tested in software as follows. At the beginning of each pass, a 1-bit per pixel test buffer was cleared. Then for each cell touched in the list, the bounding box was tested by scan converting it to check if there was any overlap with the test buffer. Because the 1 bit buffer was represented in multi-bit words, aligned per scan line to the left of the screen, the pattern along a scan line in a box often consists of a left hand partly filled word, found from a look up table, zero or more full words, and a right hand partly filled word from another look-up table. Otherwise, it consists of a single partly filled word which is the bitwise intersection of words from the left and right look-up tables. To check for overlaps, the words involved in each scan line for the box are tested against the words in the test buffer using bitwise logical *and*s, and if any of the results is non-zero, there is overlap of bounding boxes, and the cell is skipped in this pass. If not, bitwise logical *or*s are used to set the bits for the bounding box in the test buffer, and the cell is rendered into the P-buffer, and marked as completed in the list of cells. We also tried scan converting the actual projection of the cell using these logical operations, instead of just its bounding box, but the extra cost of stepping along the edges of the projection balanced the speed-up from fewer layers, and there was no net gain.

Once a threshold number of cells are skipped due to overlap, the current pass is terminated, the smallest rectangle in the P-buffer containing all the rendered bounding boxes is copied into the texture, and the next pass is started. Note that the P-buffer contains the results of adding all the passes, so when no cells are left, it is copied to the output buffer. If an X-ray image rather than an attenuation image is desired, the negative exponential is done during this copy.

The number of passes required is related to the depth complexity of the volume decomposition. This is another reason why it is preferable to render a hexahedral cell as a whole, rather than as five tetrahedra, which would increase the depth complexity.

4.1 Computing Extinction Coefficients

In order to simulate radiographs in the non-emissive regime, accurate extinction coefficients are required. In this section we describe the procedure for obtaining extinction coefficients from calculated atomic cross-sections for X-rays.

The extinction coefficient K_ν can be defined in terms of the cold opacities of the atomic elements:

$$K_\nu = \kappa\rho \tag{7}$$

where κ is the opacity in units of cm^2/g and ρ is the density of the material.

Elemental opacities can be calculated using publicly available tabulations of computed atomic cross sections [4, 7]. These tables are indexed by photon energy which is defined as $E = h\nu$ where h is Planck's constant ($6.6263 \times 10^{-27}\mathrm{erg/s}$) and ν is the frequency. The opacity for a given element can be computed from the tabulated atomic cross sections by interpolation:

$$\kappa_E = \frac{\sigma_E}{uA} \tag{8}$$

where the cross section σ_E is interpolated at energy E and has units of barns/atom ($1\mathrm{barn} = 10^{-24} cm^2$), u is the atomic mass unit (amu; $1\mathrm{amu} = 1.6605402 \times 10^{-24}\mathrm{g}$), and A is the atomic mass of the element in amu. Photon energies are typically given in electron volts ($1\mathrm{eV} = 1.60206 \times 10^{-12}\mathrm{erg}$).

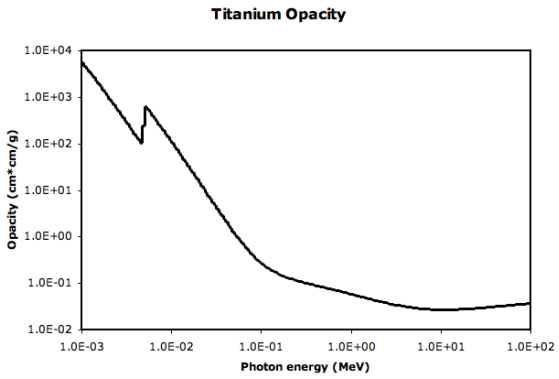


Figure 2: A plot of the opacity κ of Titanium for 200 energy groups. The sawtooth shape is due to the absorption edge of the K shell electrons.

The opacity information is divided into energy groups defined by lower and upper energies E_g and E_{g+1} respectively, where g denotes the group number and $E_g < E_{g+1}$. For each energy group a constant approximation is computed according to:

$$\kappa_g = \frac{1}{E_{g+1} - E_g} \int_{E_g}^{E_{g+1}} \kappa_{E'} \mathrm{d}E' \tag{9}$$

Figure 2 shows a piecewise constant approximation of the absorption spectrum of Titanium generated using this method. The sawtooth shape approximates the absorption edge of the K shell electrons. At an absorption edge, the photon energy passes the binding energy of an electron in a specific shell. This allows the photon to ionize the atom, and increases the opacity.

The opacities of mixtures and compounds can be estimated by assuming atomic mixing:

$$\kappa_{mat,g} = \sum_{i=0}^{N-1} m_i \kappa_{i,g} \tag{10}$$

where *mat* is a material index, g is the energy group, $\kappa_{i,g}$ is the opacity of element i, and m_i is the mass fraction of element i in the mixture or compound.

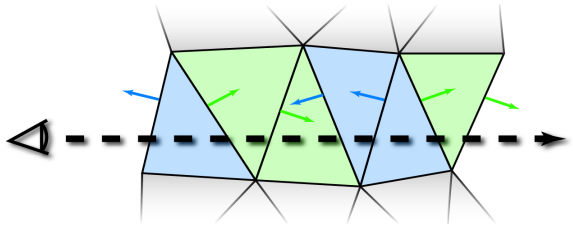


Figure 3: Unique faces share up to two cells and are encoded with extinction and source values for both. Determining which values to use (shown as blue and green cells) is determined per view by a dot product with the face normal.

The resulting extinction coefficients can be stored in a table indexed by material number and energy group number.

5 Radiography of Emissive Objects

Radiographs of emissive objects require the full solution of equation 5. Thus, the compositing operation is not commutative and must be done in back-to-front order. We assume the extinction coefficient K_ν and source term S_ν are provided as piecewise constant quantities on a geometric object or mesh. We extend the Hardware-Assisted Visibility Sorting (HAVS) volume rendering system proposed by Callahan *et al.* [2]. The HAVS algorithm operates on a list of the unique faces (triangles) in the tetrahedral mesh. In object-space, the faces are depth sorted by their centroid and then passed to the GPU for rasterization. This first sorting pass places the geometry in an approximate depth order. Upon rasterization, the fragments undergo an image-space sort via the k-buffer which has been implemented on programmable graphics hardware using fragment shaders. The k-buffer is a fragment stream sorting network that keeps a fixed number (k) of fragments for each pixel in the framebuffer. As a new fragment is rasterized, it is compared with the other entries in the k-buffer and the two entries that are farthest from the viewpoint (for back-to-front traversal) are used to find the color and opacity for the fragment using a lookup table which contains the preintegrated volume integral [14]. The color and opacity are composited into the framebuffer, and the remaining fragments are written back to the k-buffer.

In our case, we are interested in solving the transport equation for each energy group along a ray passing though a set of ordered cells. Unlike the previously described approach, a pre-computed lookup table would introduce considerable error due to the large dynamic range of extinction coefficients and source terms encountered in many applications. Therefore, we compute the transport equation in the fragment program which requires the opacity value and source function value for each energy group in addition to the distance of the fragment from the viewpoint. We limit the number of energy groups we visualize to four, one per image channel (R,G,B,A). Since the HAVS algorithm uses Multiple Render Targets (MRT) to keep the k-buffer entries during rasterization, the size of k depends on the number of values that are stored for each fragment. In the original algorithm, HAVS only uses two entries (v,d) per fragment: one for scalar value and the other for distance from the viewpoint. On current hardware, three render targets are available in addition to the framebuffer which allow up to six fragment entries ($k = 6$). Unfortunately, by removing the lookup table, we require much more information in our per-fragment computation. The viewpoint distance for each fragment is still needed as well as an opacity value and a source function value per energy group. Storing all these values for each fragment would only allow one fragment entry in the k-buffer. This would limit the amount of image-space sorting that occurs and may affect overall image quality. A more conservative approach is to perform a separate rendering pass for each energy group g. This allows each fragment entry to be reduced to three values, the distance, opacity, and source for the current energy group (d,o_g,s_g), which leads to a k size of four. This improves the image quality by sacrificing some performance.

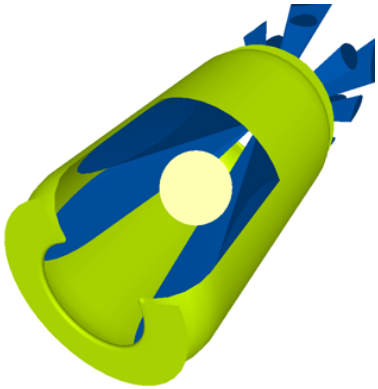
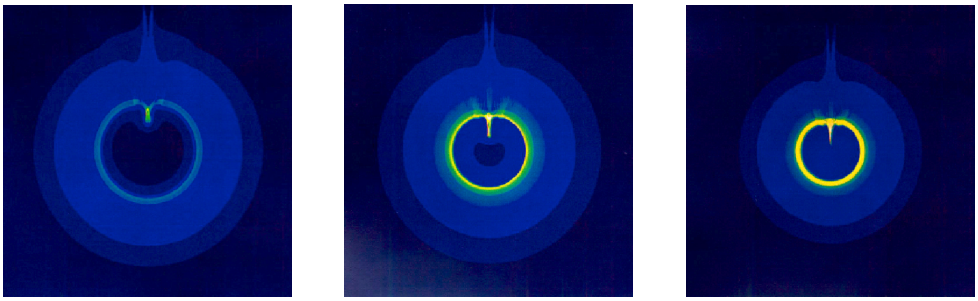


Figure 4: An overview of the ICF process. Several laser beams strike the inside surface of a hohlraum (the green cylindrical canister) and heat it to millions of degrees. The resulting thermal X-rays emitted by the hohlraum's inner surface heat the outer surface of the capsule. The "rocket exhaust" effect of material ablated from the capsule's surface compresses the hydrogen fuel in the center to conditions where fusion occurs.

Another important detail to consider is that our transport equations provide a first order recurrence over a piece-wise constant approximation. Thus, the opacity and source terms for the energy groups are assigned per cell. Since the HAVS algorithm operates on the unique faces that make up a tetrahedral mesh, no connectivity information is maintained, unlike in ray-casting approaches [17, 18]. This also considerably reduces the number of primitives that are rasterized when compared with the Projected Tetrahedra [16] algorithm. To keep the fragment count low, these unique faces are extended to keep the energy group terms for possibly two cells (the cell that the face belongs to and a neighboring cell if it is not on the boundary).

Figure 3 shows how opacity and source terms are determined using only face information. A ray passing through the volume will enter a cell by passing through one face and exit the cell by passing through the next face. To determine whether a ray is entering or exiting a cell represented only by a face, a comparison can be made of the normal direction of the face and the direction of the ray. In an extra object-space step, we perform a dot product of every face normal with the viewing direction and choose the energy groups for the faces based on the view. Another possibility we would like to explore is to store the two energy groups in material properties of the primitives and decide at the fragment level which energy group to use. This would exploit recent fragment program extensions that specify whether a front or back face is rasterized.

A final consideration is that due to the nature of graphics hardware, the extinction coefficients and source terms are passed to the GPU per vertex in texture coordinates. However, since multiple faces may share one vertex, this can cause inconsistencies in the energy group terms across a face. Therefore we perform a preprocessing step which duplicates the vertices that are shared by multiple faces. Though this introduces some redundancy in the vertices, the number of rasterized faces remains the same and constant source and opacity values are guaranteed across the faces.



17**ns** 17.05**ns** 17.0875**ns**

Figure 5: A simulation showing a jet formed inside the fill tube as the capsule implodes. Times are shown in nano-seconds.

6 Application: Designing Diagnostics for NIF Experiments

We now describe an application involving radiographs of subjects with emissive materials for laser experiments. When completed, the National Ignition Facility (NIF) is expected to produce fusion by compressing a tiny capsule of Deuterium-Tritium (DT) fuel in a process known as inertial confinement fusion (ICF). NIF experiments will reproduce the physical conditions inside stars.

Figure 4 depicts the indirect-drive ICF process. A small spherical capsule with a radius of roughly 1 millimeter is placed inside a hollow cylinder called a hohlraum (a German word meaning "cavity") which is typically a few millimeters in diameter and about 10 millimeters long. Multiple laser beams enter through the ends of a hohlraum and strike its inside surface. The hohlraum is composed of a heavy material like gold or lead which is heated to millions of degrees by the laser beams. The hohlraum wall is hot enough that it emits X-rays which bathe the inside of the hohlraum and cause the ablation of the outer (plastic) surface of the capsule. This ablation turns the outer surface of the capsule into a spherical rocket, driving the shell of the capsule inward and compressing the DT fuel within.

Two key components of the ICF process are that the DT fuel remains cool during the compression process and that the compression is uniform (spherically symmetric). If the fuel is too warm, the DT will be more difficult to compress and the fusion output will be reduced. If the compression is not symmetric, the peak density will be lowered, again resulting in a smaller amount of fusion.

Capsules will be filled with fuel through a tiny tube connecting the capsule to a reservoir outside the hohlraum. Simulations and experiments of this system have shown that a plasma jet is formed by the tube as it is imploded along with the capsule by the X-rays. The presence of the jet has been detected in experiments on a smaller laser at the University of Rochester. Follow-up experiments are necessary to accurately measure the position and velocity of the jet and compare this information against simulations. The primary goal of this paper is to demonstrate GPU based radiography algorithms that enhance the productivity of scientists designing such experiments. It is likely that after a set of experimental parameters have been determined, a scientist will validate the results with a full precision software calculation. Thus, our contribution is a drop-in replacement for software approaches that trades off some accuracy for increased performance.

6.1 Imaging the Effects of the Fill Tube

Figure 5 shows the jet that is formed by the fill tube as the outer layer of the capsule is ablated. A fundamental question is the effect of this jet on capsule implosion. Scientists are designing experiments using capsules that have shells doped with titanium. When these doped capsules are imploded the titanium dopant emits X-rays at a higher intensity than the capsule material. The diagnostics must be designed to maximize the photon count at the detector, and to enhance the contrast between the dopant (Ti) and the shell material (primarily carbon). Simulated radiography is used optimize the free parameters.

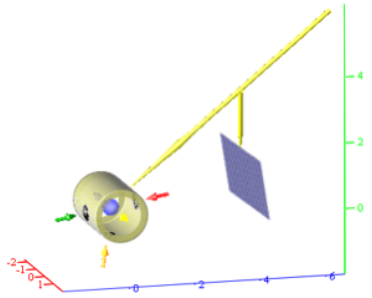


Figure 6: X-rays emitted from the capsule travel through the hohlraum and onto an array of pinholes.

Figure 6 shows the set up of the experiment.

There are several parameters that can be adjusted to achieve these goals:

1. **Capsule Dopant:** changing the dopant and varying the amount of dopant in the capsule. The jet must be detected along with the capsule material in order to discover the position of the jet relative to the capsule.

2. **Reference Wires:** adding one or more wires of an optically thick material to the outside of the hohlraum. The wires would cast shadows on the detector to enable accurate measurement of the position of the jet. The thickness and spacing of these wires is critical.

3. **Filtering of X-rays:** introducing slabs of different materials between the hohlraum and detector can cause preferential attenuation of X-rays at certain wavelengths. These can be used to emphasize emission of one material over another. However, the use of additional filtering reduces the overall intensity of the radiation striking the detector, resulting in more noise in the images.

The application presented in this paper requires images of about 160 by 160 pixels where the resolution is limited by detector sensitivity. Actual detectors have uncertainties in calibration, limits on spatial and temporal resolution, and are subject to noise. Scientists simulating radiographs with current software methods do not expect predictions to be accurate to closer than 10% but systematic

errors are not acceptable. We focus on performance gains produced by using graphics hardware to solve the transport equation. Our goal is to render images at speeds high enough that the process of adjusting experimental parameters above becomes less onerous.

7 RESULTS

There is a large variation in requirements for radiographic diagnostics. Detector/film resolutions vary from a few hundred pixels per side to several thousand. Cell projection methods shine on large images, but can also be applied to small images with appropriate trade-offs. The scientists we interact with uniformly use raytracing algorithms in both industrial and high-energy physics regimes. We expect our techniques to show the most utility at large image sizes, but below we present results that are competitive even for the small detector resolution of the capsule experiment.

7.1 Attenuation-Only Radiographs

Figure 7: A simulated radiograph using low intensity X-rays. The capsule and hohlraum material reduce the received intensity, but the ejected material is visible.

Figure 8: Higher intensity radiation resolves the capsule.

Figures 7 and 8 show absorption-only radiographs of a simulated hohlraum and capsule using the unsorted layer-rendering algorithm with a low and high intensity X-ray source. This data set contained five different materials for which extinction coefficients were computed based on the EPDL [4] database. The detector was assumed to have uniform sensitivity at all photon energies.

This data set had 32 curvilinear grid domains, for a total of 141,960 hexahedral cells, of which 40,656 were collapsed to a single point. There were also 9696 other cells which had at least two coinciding vertices, and 924 with quadrilateral faces with non-convex projections. Cells of both types were subdivided into tetrahedra and projected by the method of [16]. The remaining 90,684 hexahedra were projected by the method of Max [12]. On one processor of a dual processor 860 MHz Pentium4 Xeon PC, with an nVidia 6800 GT graphics card, it took 2.2 seconds to read in the data and compute an attenuation table for four frequency bands and the five materials which may be mixed into each cell. Another 35 seconds were required to determine the overlaps in software, render the cells is 824 separate layers into the P-buffer and copy them back out into the texture, and finally draw the texture to the screen at 512×512 resolution.

The overlap determination and texture copying are a bottleneck to this method. One way to eliminate this bottleneck is to read and write from the same framebuffer object, as in [9]. As discussed above, this can cause errors if texture data is used before it has been written out. We experimentally found that we had to draw a textured square of size 120×120 pixels to remove these read-before-write artifacts. Nevertheless, the rendering time decreased from 35 seconds to 7.4 seconds.

7.2 Attenuation Plus Emission Radiographs

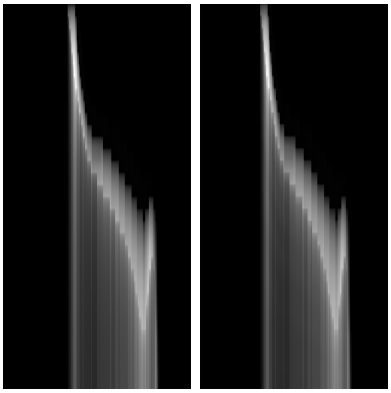


Figure 9: (Left) Software raytrace of gas bag experiment showing the 10th energy group on a 321×33 grid with cylindrical symmetry. Only the top half of the image is generated. (Right) HAVS image of the same grid extended to three dimensions by rotation axially. The final mesh size was $321 \times 33 \times 16$ vertices.

The HAVS algorithm was validated against the two raytrace packages currently used by scientists for simulation of radiographs of laser-plasma experiments. A 2-D raytracer is used for cylindrically symmetric meshes and is optimized for that case. A 3-D raytracer is used for hexahedral meshes. The 3-D package divides hexes into five tetrahedra to compute ray propagation. The three techniques were used to produce radiographs of a 'gas bag' data set. In this experiment, a cylinder is filled with a cold gas of various elements and a laser beam is directed down the axis of the cylinder. A radiographic diagnostic is used to visualize the laser-plasma interaction.

Figure 9 shows the radiographs generated by the 2-D software raytrace and the HAVS algorithm. The initial 2-D mesh had an arrangement of 321×33 vertices with 12 energy groups. The data was extruded 180 degrees about the axis of the cylinder to match the 2-D software raytrace. The final 3-D mesh size was $321 \times 33 \times 16$ vertices. The HAVS and 3-D software implementations were applied to this mesh.

Table 1: Solution times in seconds for gas bag data set

Image Size	2-D Raytrace		3-D Raytrace		HAVS
	860MHz	2.2GHz	860MHz	2.2GHz	860MHz
128×128	6.1	2.5	44	18	17
256×256	23.4	9	216.4	81.2	17

Table 1 shows times in seconds for two different image sizes obtained on two different workstations. All algorithms were run on a 860 MHz Pentium Xeon Linux workstation with an NVidia 6800 GT. The software packages were also tested on a 2.2GHz Xeon workstation. The results show that depending on the resolution used to create the extrusion the efficiency cross-over for the hardware implementation occurs at image sizes of about 256×256 for the 2-D raytrace, and at 128×128 for the 3-D raytrace. The raytrace packages are linear in image size while the HAVS algorithm timings remain nearly constant until the card memory limit is reached. Interestingly, we found that the software implementations scale linearly in the number of energy groups, which allows the HAVS implementation to be competitive even for large numbers of energy groups. Experience with the HAVS algorithm for visualization applications indicates that its performance can vary between ATI and NVidia hardware. We have found that on an ATI X800 Pro our algorithm can run up to two times faster than on an nVidia 6800 GT.

The software raytracer produced radiographs with intensities ranging from 5.6×10^{-15} to 1.4×10^{-3}. We compared the intensities computed by 2-D software raytrace against the HAVS algorithm for an image size of 512×512. We observed different error characteristics depending the absolute intensity of pixels. In low-emission regions (intensity less than 10^{-8}) of the image the HAVS algorithm had a mean relative error of 11 to 30 percent with a maximum of 98%. These errors appeared evenly spread across low-emission regions of the radiograph. We suspect these errors are due to a combination of small extinction and source terms and 32-bit arithmetic. The HAVS algorithm was substantially more accurate for pixels with intensity greater than 10^{-9} with a mean relative error of 1.6 to 2.5 percent and a maximum of nearly 95 percent. The large errors are due to pixel size differences in the vertical locations of horizontal edges in the intensity introduced by the piecewise linear extrusion of the 2-D mesh about the axis of symmetry. Many of these edges were jumps in intensity by an order of magnitude across two or three pixels, so pixel-by-pixel comparisons showed a large relative error at these locations. Finally, we expect some errors to occur due to incorrect fragment ordering which can occur when the k-buffer algorithm is used.

7.3 Simulated Radiographs of Capsule Experiments

Figure 10 depicts the experimental setup for detection of the jet created by the capsule fill tube. The capsule material is doped with titanium which is more emissive than the shell material which is primarily plastic. The X-rays pass through a window cut into the hohlraum made of a lower Z material such as beryllium (Z=4). The X-rays pass through an array of pinholes, which are imaged at different times to generate a time-lapse series to capture the compression of the capsule.

This apparatus is simulated in several stages. First, the capsule is simulated at high resolution ($\sim$ 10 million cells) by a coupled radiation-hydrodynamics code. The simulation models the progress of the compression and the jet formed by the fill tube. The result of the simulation is a mesh defining the emissivity and opacity per cell per photon energy group. This data is used as input to a raytracing package that computes the photon intensity over each detector pixel

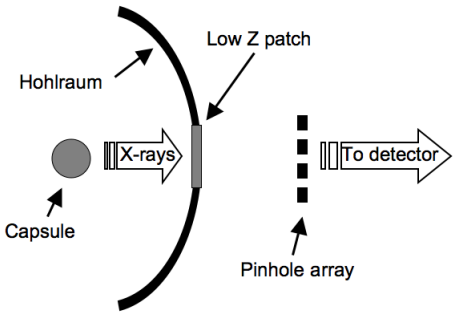


Figure 10: The diagnostic set-up depicting each stage of simulation. The heated capsule shell emits X-rays, which travel through a window on the hohlraum made of a material with lower atomic number than the hohlraum itself. The radiation then passes through an array of pinholes, which are imaged in successively to create a set of images in time. The GPU-based tetrahedral projection code is used to simulate the initial set of radiation leaving the hohlraum.

for each photon energy group in the simulation. The effects of the low Z window, the pinholes, and any other filters are computed using image processing techniques on the resulting photon intensity information computed by the raytrace. For example, a Gaussian blur is used to model pinholes and noise is added directly to the image in a single pass. These image processing operations account for less than 5% of the processing time.

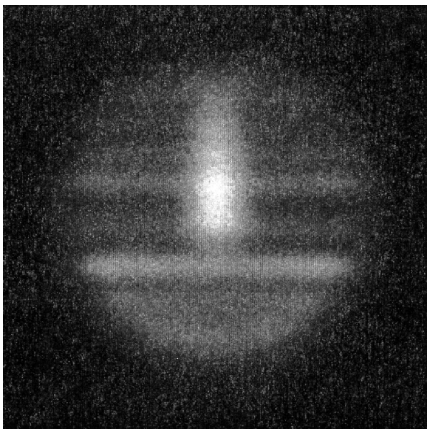


Figure 11: A simulated radiograph of the fill-tube experiment creating using the raytrace package used by the physicists. The original simulated data is two dimensional, so all features that are not on the axis are rotated about the axial coordinate creating the banding. The jet created by the fill tube is visible on the Z-axis (vertical).

The fill-tube simulation resulted in a 2-D mesh of 47,750 quadrilaterals with assumed cylindrical symmetry. The software raytrace used by the experimentalists raytraces the cylindrical extrusions of each quadrilateral in the data set. For the hardware-accelerated k-buffer algorithm the entire data set was extended about the axial coordinate, partitioned into hexahedra, then partitioned into tetrahedra.

Figure 11 shows a simulated radiograph of a fill-tube jet experi-

ment. The image size was 160×160 pixels. The software raytracing package currently in use by the experimentalists took 2 minutes for 57 energy groups on a machine with a 2.2 GHz Pentium class processor and 896 MB of RAM. The hardware accelerated k-buffer algorithm was applied to a 3-D stand-in of the fill tube data with $1,719,000$ tetrahedra. The system was a 3.2 GHz Pentium 4 with 2 gigabytes of RAM running Windows XP. The graphics card was an ATI X800 Pro with 256 MB of RAM. The initial sort took 0.656 seconds and the solution for all energy groups took 57.7 seconds. We validated the results by running a software emulator of the k-buffer algorithm on the actual fill tube data set.

8 Conclusion and Future Work

We have demonstrated modified volume rendering algorithms that can be used to produce accurate simulated radiographs. We described implementations of cell projection algorithms with and without sorting for the attenuation-only and attenuation-emission cases. The algorithms described require recent graphics cards and drivers, due to the requirement of floating point accuracy and the use of advanced texture capabilities.

Our goal of interactivity for users at their workstations will require modification of the algorithms. Accuracy must be carefully traded off against performance. A combination of smoothing and coarsening of meshes, a reduction in the number of photon energy groups by preintegration, and more precomputation will be required. It is possible that using 16-bit framebuffer capabilities may also be an option for some applications. We would like to be able to produce a visualization tool that enables interactive subsetting, orientation, and modification of material properties so that scientists who combine simulation with experiment can better understand their data.

In the polyhedron projection method of Section 4.1, which is guaranteed to be accurate because it does not involve read-before-write hazards or limitations in the k size of the HAVS k-buffer, the software scan conversion is a speed bottleneck. The Meshed Polyhedra Visibility Ordering (MPVO) sorting algorithm [19] can also produce layers of non-overlapping cells, if the data volume is convex, and the grid connection topology (which cells share faces) is known. In our application to curvilinear grids (see Section 7.1) we currently do not know the connection topology across the multiple domain boundaries, but hope to reconstruct it in the future. For a non-convex data volume, the Scanning Exact MPVO (SXMPVO) sort of Cook *et al.* [3] uses software scan conversion of only the external faces to add the necessary extra links across empty gaps to the directed visibility graph. A breadth-first sort of this graph can produce a set of non-overlapping layers for even a non-convex mesh, and we plan to apply this in the future to attenuation plus emission radiographs, where sorting is required. We also plan to investigate parallel implementations on graphics clusters to obtain interactive performance. Several strategies are possible including distributing photon energy groups across several processors in cases where the data can be duplicated and there are many photon energies. Domain decomposition is a trivial exercise for absorption-only radiographs thanks to order independence. However, domain decomposition for algorithms modeling emissive materials will require modification due to possible ordering complications at domain boundaries.

9 Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48 (UCRL-CONF-211396). Steven P. Callahan and Cláudio T. Silva are supported under grants from the DOE and NSF. The authors would like to thank Holger Jones, Chris Wynn, and Jeremy Zelsnack for help getting started with P-buffers and framebuffer objects.

References

[1] A. Bonin, B. Lavayssiere, and B. Chalmond. MODERATO: a monte-carlo radiographic simulation. In *Proc. Review of Progress in Quantitative NDE*, july 1999.

[2] Steven P. Callahan, Milan Ikits, Joao L.D. Comba, and Cláudio T. Silva. Hardware-assisted visibility ordering for unstructured volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):285–295, 2005.

[3] R. Cook, N. Max, C. T. Silva, and P. Williams. Efficient, exact visibility ordering of unstructured meshes. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):695–707, 2004.

[4] Dermott E. Cullen, John H. Hubble, and Lynn Kissel. EPDL97: the evaluated photon data library, '97 version. Technical report, UCRL-ID-50400, Lawrence Livermore National Laboratory, 1997.

[5] P. Duvauchelle, N. Freud, V. Kaftandjian, G. Peix, and D. Babot. A computer code to simulate x-ray imaging techniques. *Nuclear Instruments and Methods in Physics Research B*, 170:245–258, 2000.

[6] A. Glière. Sindbad: From CAD model to synthetic radiographs. *Review of Progress in Quantitative Nondestructive Evaluation*, 17:387–394, 1998.

[7] J. H. Hubble and S. M. Seltzer. Tables of x-ray mass attenuation coefficients and mass-absorption coefficients. Technical report, [Online], National Institute of Standards and Technology], Gaithersburg, MD, 2004. Originally published as NISTIR 5632.

[8] A. Koenig, A. Gliere, R. Sauze, and P. Rizo. Radiograph simulation to enhance defect detection and characterization. In *"Proc. of the* 7^{th} *European Conference on Non-Destructive Testing"*, volume 1, pages 444–451, 1998.

[9] Martin Kraus, Wei Qiao, and Davis S. Ebert. Projected tetrahedra without rendering artifacts. In *Proceedings of IEEE Visualization*, pages 27–34, 2004.

[10] Jens Kruger and Rudiger Westermann. Linear algebra operators for GPU implementation of numerical algorithms. *ACM Trans. Graph.*, 22(3):908–916, 2003.

[11] Youquan Liu, Xuehui Liu, and Enhua Wu. Real-time 3D fluid simulation on GPU with complex obstacles. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference on (PG'04)*, pages 247–256, Washington, DC, USA, 2004. IEEE Computer Society.

[12] Nelson Max. Hexahedron projection for curvilinear grids. Poster at volume graphics 2005, LLNL Tech. Report 318089, 2005.

[13] Nelson max, Peter Williams, and Cláudio Silva. Approximate volume rendering for curvilinear and unstructured grids by hardware-assisted polyhedron projection. *International Journal of Imaging Systems and Technology*, 11:53–61, 2000.

[14] S. Roettger, M. Kraus, and T. Ertl. Hardware-accelerated volume and isosurface rendering based on cell-projection. In *Proceedings of IEEE Visualization*, pages 109–116, oct 2000.

[15] Stefan Roettger and Thomas Ertl. Cell projection of convex polyhedra. In *Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics*, pages 103–107, 2003.

[16] Peter Shirley and Allan Tuchman. A polygonal approximation to direct scalar volume rendering. *Proc. San Diego Workshop on Volume Visualization*, 24(5):63–70, nov 1990.

[17] Manfred Weiler, Martin Kraus, Markus Merz, and Thomas Ertl. Hardware-based ray casting for tetrahedral meshes. In *Proc. IEEE Visualization*, pages 333–340, oct 2003.

[18] Manfred Weiler, Paula N. Mallón, Martin Kraus, and Thomas Ertl. Texture-Encoded Tetrahedral Strips. In *Proceedings Symposium on Volume Visualization 2004*, pages 71–78. IEEE, 2004.

[19] Peter L. Williams. Visibility-ordering meshed polyhedra. *ACM Transactions on Graphics*, 11(2):103–126, apr 1992.

[20] J. Xu, R. Wallingford, T. Jensen, and J. Gray. Recent developments in the x-ray simulation code: XRSIM. volume 13a, pages 557–562, 1994.