

A Hardware-Assisted Hybrid Rendering Technique for Interactive Volume Visualization

Brett Wilson* Kwan-Liu Ma[†]
Computer Science Department
University of California at Davis

Patrick S. McCormick[‡]
Advanced Computing Laboratory
Los Alamos National Laboratory

ABSTRACT

The scientific simulation and three-dimensional imaging systems in use today are producing large quantities of data that range from gigabytes to petabytes in size. Direct volume rendering, using hardware-based three-dimensional textures, is a common technique for interactively exploring these data sets. The most serious drawback of this approach is the finite amount of available texture memory. In this paper we introduce a hybrid volume rendering technique based on the use of hardware texture mapping and point-based rendering. This approach allows us to leverage the performance of hardware-based volume rendering and the flexibility of a point-based rendering to generate a more efficient representation that makes possible interactive exploration of large-scale data using a single PC.

Keywords: interactive visualization, large data visualization, point-based rendering, texture graphics hardware, volume rendering

1 INTRODUCTION

Many scientific computations and three-dimensional imaging systems generate data portraying physical phenomena or structures in three-dimensions. Direct volume rendering is a common technique used for visualizing this type of data. It is straightforward and preserves all the features of the original data, including structures that can not be analytically defined. With the introduction of hardware-accelerated volume rendering [23, 13], these data sets can be rendered and explored at interactive rates. However, the amount of available texture memory and the fill rate of the graphics system limit the size of the data that can be processed efficiently in this manner. Large data sets also present challenges with respect to storage and network capacity. In this paper, we present a hybrid rendering technique designed to allow the interactive visualization of large volumetric data with low-cost consumer graphics cards such as the nVidia GeForce 4 and commodity PCs with limited network, storage, and memory capacity.

Our hybrid technique leverages the speed of texture-based, hardware-accelerated volume rendering and the flexibility of point-based rendering. The volumetric data is used to represent large, smooth features and is augmented with point data in areas of fine detail or fast change. This approach allows a more efficient allocation of storage and rendering resources than either volume rendering or point-based rendering alone, while producing limited artifacts.

*wilson@cs.ucdavis.edu

[†]ma@cs.ucdavis.edu

[‡]pat@lanl.gov

We do not intend to completely replace high-accuracy volume rendering with this hybrid method. Rather, this new approach allows us to more efficiently store, transfer and interactively preview reasonably accurate data using a single low-cost PC and graphics card. More importantly, the size reduction the hybrid method provides allows real-time interaction with data too large to display on a given computer, while maintaining as much detail as possible.

2 RELATED WORK

The use of points as a display primitive was first introduced in [8] and was more recently presented in [4]. More closely related to our effort are the point-based rendering methods used to handle the interactive display of large and complex polygonal data sets [18, 20]. In the context of volume rendering, splatting [22] has become an increasingly popular approach. We have seen the development of faster [7] and higher-quality [19] splatting techniques, the use of splatting for rendering irregular-grid data [11, 12], a more universal approach in reconstructing surface and volume data [24], and a hybrid point rendering method for rendering particle beams [10] that uses volumetric data to represent high-density regions.

Hardware-accelerated volume rendering is usually implemented using either two-dimensional [1] or three-dimensional [23] texture-mapping hardware. Dedicated volume-rendering hardware [13] has also been used. However, even on the most advanced systems, the size of many data sets can far exceed available video memory. There are several approaches to rendering large volumetric data that does not fit within the video card's memory. The simplest is texture paging, which is often implemented in the display driver and is used automatically when memory is full. Unfortunately the latency required for transferring data over the bus to video memory inhibits interactive display rates. Past efforts have also used parallel rendering techniques to increase the amount of available texture memory [9, 5]. While this approach is feasible, the required amount of hardware is often costly, efficient implementation can be complex, and the system is still limited by a fixed amount of texture memory. Finally, multi-resolution techniques [21, 6] attempt to make better use of texture memory by using lower sampling rates for areas of low interest. The disadvantages of this technique are the introduction of artifacts and loss of detail in the lower-resolution areas.

3 HYBRID DATA GENERATION

Converting a high-resolution volume data set to a hybrid representation involves generating a low-resolution volume and a corresponding set of points. The low-resolution volume data represents the larger, more uniform areas of the original, and the points are inserted where the low-resolution data skips or de-emphasizes important details. Thus, storage and processing techniques are allocated according to the detail requirements of a given region of data. This conversion is done by a preprocessing program that requires on the order of tens of minutes for typical data sets on a powerful desk-

top PC. We have also had success preprocessing portions of data on smaller computers and merging the resulting hybrid representations (this technique could be used to efficiently preprocess data in parallel).

Any standard subsampling technique can be used to generate the low-resolution volume data; the best choice depends on the nature of the data and the application it is used for. Most images in this paper were generated using cubic interpolation, although other subsampling techniques are more appropriate in certain cases (see "Optimized Representations" below).

3.1 Point Selection

Points are generated at those locations where the volumetric data contains large amounts of error. These locations are identified by sampling the difference between the low-resolution representation and the original data at regular intervals. Typically, the sampling interval of the error grid is the same as the original resolution of the data, meaning the low-resolution data is compared to each of the original data values. Lower-resolution error grids can also be used. Linear interpolation is used on the low-resolution data to simulate the value the video card will compute at each error grid location. Typical point insertion is done based on a threshold error value: if the error value is above a user-defined level, a point is inserted at that location. If, a maximum output size is desired, a given number of points can be inserted at the locations with the highest error magnitude.

3.2 Optimized Representations

One source of error results from the limitation that video cards can only add color on the screen. That is, points can only contribute to the opacity of a given area, and never make it more transparent. On average, a low-resolution representation will overestimate the original function as much as it underestimates, so the addition of points can only make up 50% of the error in the best case.

If something is known at the time of preprocessing about the type of transfer function the user will select, the error can be improved by selecting the low-resolution representation such that it never produces positive error for the expected transfer function. One common type of transfer function is one that is never decreasing; the user often wishes to see areas of higher value as always being more opaque. In this instance, the low-resolution representation is selected so that it never overestimates the original function value (the interpolation function is the minimum function). In practice, we found this method produces poor results. It generates large areas where the original function is extremely underestimated, so it requires an impractical number of points to produce a good image. As a compromise, our preprocessing program can take a parameter that specifies the minimum amount the low-resolution representation can overestimate the original, allowing the user to decide the extent to which this type of accuracy balances the number of points.

3.3 Storage

The low-resolution volumetric data and point data are calculated in a preprocessing step and stored for later viewing. The sampling interval and error threshold are provided as parameters to the preprocessing program. For points, the position, error value, and the value and normal of the original data at that location are stored. The original function value is used to map the point to the corresponding color and opacity when it is displayed. The error value indicates how much the opacity should be scaled so that, when combined with the volumetric data, the correct opacity is presented on the screen. The preprocessor also computes normals for each value of

the volumetric data. All normals are stored as one-byte indices into a lookup table containing normal vectors uniformly distributed in 3-space.

If a 256^3 grid is used for error calculation, each point can be stored with 8-bit coordinates, giving a storage space of 6 bytes (one byte each for x, y, z, normal, error, and original function value). If a higher-resolution grid is used, each point requires 16-bit coordinates, giving 9 bytes of storage per point. An optimized hierarchical representation such as that presented in [18] could reduce the storage required for coordinates to about 16 bits for each point, giving 5 bytes storage per point. We did not implement this method because the data size is dominated by volume data and the corresponding normals, and such optimization would only reduce the total data size by 5–10% for the examples presented in this paper.

A 512^3 volume with one-byte normals can therefore be converted into a 256^3 volume with enough room for 26 million points in the same amount of file space (46 million using the 5-byte hierarchical coordinate representation). In practice, many fewer points are required to achieve good results, and even lower resolution volumes are acceptable for some purposes. Therefore, the hybrid approach can give a significant savings in disk and memory requirements over the non-hybrid method.

4 RENDERING HYBRID DATA

The volumetric portion of the image is rendered using parallel, axis-aligned, texture-mapped planes. Each texture is an indexed-color slice of the data, allowing fast updating of the transfer function by manipulating the palette. To prevent the volume from disappearing when the slices are viewed edge-on, three sets of slices are maintained, one for each axis, and the set most parallel to the view plane is used. Since each slice of the data is usually partially transparent, the video card's Z-buffer will not correctly handle ordering of objects. Therefore, special care must be taken to draw items back-to-front.

Each slice also has a corresponding normal texture map. This map contains the one-byte normal indices computed during preprocessing, stored as an indexed texture. A palette maps these indices to the specular (alpha channel) and diffuse (red, green, and blue channels) lighting values for the corresponding normal vector. Multitexturing and hardware register combiners are used to multiply each data value with its diffuse lighting value, and then add the specular light value.

Slices of points are drawn interleaved with each pair of volumetric planes so ordering is maintained. These points can be rendered using either an OpenGL point primitive, which is rendered as a square, or a Gaussian splat. OpenGL points are sized such that adjacent points overlap slightly (Gaussian splats must overlap more since their opacity falls off toward the edges). If adjacent points do not meet, distracting patterns will appear, and if they overlap too much, the covered area will be more opaque than was intended during preprocessing. Since OpenGL points are drawn with integer pixel sizes only, problems can arise when the correct point size is a small non-integer. In these cases, the roundoff error is a large fraction of the correct value, resulting in points that are much too big or too small. To combat this problem, point sizes are always rounded up when passed to OpenGL, and their opacity is adjusted downward proportional to the increase in area of the point.

For greater rendering efficiency when drawing OpenGL points, each slice of points between adjacent pairs of planes are loaded into a display list. This triples the amount of memory required for point storage on the video card, since each point is loaded into one display list for each axis, but yields greater rendering efficiency: even if the video driver must page the display list data out of video memory, it is still faster than loading the point data from scratch. Gaussian splats are rendered using a Gaussian texture mapped onto

forward-facing squares (billboards). Since rendering a forward-facing primitive is a view-dependent operation, Gaussian splats can not be loaded into display lists, increasing rendering time.

No care is taken to render points within a slice back-to-front, and this can introduce artifacts [8]. Although a hardware-based solution is given in [18], it increases rendering time, and the goal of our hybrid method is to display a reasonably accurate representation of large volumes as fast as possible. If a more accurate representation is desired, a more effective use of display resources would be to increase volume resolution or point counts. Furthermore, artifacts are limited for two reasons. First, whereas the previous work uses points to display primarily opaque surfaces, our points are mostly transparent and ordering is less of a factor in the combined result. Second, since the error sampling grid is only a small factor (usually two to eight) greater than the volumetric slice resolution (the scale at which ordering is correct), the number of points with incorrect ordering is limited to the same small factor.

The video memory requirements of the texture data could be reduced by using the volumetric texture support of recent video cards, which would eliminate the data duplication caused by maintaining three sets of planes for the volume. Unfortunately, most video cards do not support texture paging for portions of a volumetric texture, preventing the use of textures larger than video memory. This means a maximum volumetric texture size of 256^3 on the nVidia GeForce4 used for this project. Nevertheless, volumetric textures are still desirable because the parallel planes drawn on the screen do not have to be aligned on the texture's axis. The planes can be always be drawn parallel to the view plane while the texture coordinates are transformed, and the video card will interpolate the texture in three-dimensions, reducing rendering artifacts [23]. However, this method greatly complicates correct ordering when combined with point rendering, since arbitrary slices of points must be selected to interleave with each texture-mapped plane. It also prevents the use of display lists for point slices, since the shape of each slice of points is view-dependent. As a result, we chose to stay with axis-aligned rendering planes for better speed, ease of implementation, and to be able to generate large (512^3) hardware-accelerated reference renderings.

4.1 Sources of Error

Although the hybrid method can make up considerable amounts of error present in lower-resolution volume representations, it is not without its own sources of error. First, as was discussed above, points are only additive and can not compensate for overestimation errors. Second, if the transfer function's color map is rapidly changing, or error is great, the volumetric data will map to a significantly different color than the correct value represented by a point at that location. The rendering algorithm described above only gives correct opacity for the combination of a point and volume, the color may be incorrect (more precisely, it will be the average of the correct color and the color of the volume data). Last, since the hybrid method is essentially a multi-resolution rendering method, it is subject to the same sources of error. In areas displayed at higher resolution through the use of points, the transfer function must be adjusted downward to give consistent opacity over the entire image. In addition, since the points tend to exist on edges and corners, the amount of adjustment required is highly view-dependent. We chose not to address the view-dependent multi-resolution problem in this current work, and found that global point transparency can be manually adjusted to achieve acceptable results in many cases.

5 RESULTS

We compared hybrid renderings generated with different parameters with standard volumetric representations of the same data. All tests and timing measurements were done on a 1.0 GHz Pentium 3 Xeon running Linux with 1 GB RAM and an nVidia GeForce4 Ti 4600 graphics card with 128 MB of video memory.

5.1 The Argon Bubble Data Set

The argon bubble data set is one frame of a $640 \times 256 \times 256$ time-varying simulation of an argon bubble. It was padded and converted to 512^3 using tricubic interpolation. This step is required for an accurate reference, since most video cards require texture sizes to be powers of two. The hybrid representations and lower-resolution volume representations were all generated from the 512^3 version of the data.

The argon bubble contains some very fine detail in the turbulence, as well as many localized areas of very high density surrounded by areas of low density. As can be seen in in Figure 1, a 256^3 volume rendering misses some details, produces incorrect density values for some small features (for example, the small protrusion at the top-center) and exhibits severe pixelation artifacts. Using this low-resolution volume, a hybrid representation was generated using an error calculation grid of 512^3 and an error threshold of $1/64$, producing 361K points. The size of the hybrid representation is a mere 37MB, compared to 33MB for the low-resolution volume rendering alone, and 268MB for the original.

The hybrid renderings, even with this relatively low point count (the later examples use several million points) is able to improve the low-resolution rendering in several respects. Pixelation artifacts in key areas are decreased, and shapes are more obvious. It also corrects color in some areas, such as the small yellow feature in the top center, which appears as red in the low-resolution version. Notice that an artifact of point rendering, particularly visible at this extreme magnification, is that small details tend to get larger (this hybrid volume was not produced using the optimized representation given above to reduce the effect).

The hybrid rendering drawn with OpenGL points in the lower-left of Figure 1 display interactively (10 fps), but their square shape is clearly visible when zoomed in. The Gaussian splats in the lower-right image look much smoother, but take several seconds to render. One approach would be to use OpenGL points for normal interaction, and use Gaussian splats when the user pauses as the zoom level is high.

5.2 The Chest Data Set

The chest data set is a 512^3 MRI scan of a man's chest. At full resolution, the fine airway structure of the lungs is clearly resolved. However, at lower resolutions, almost all of this detail is lost. The hybrid approach is able to restore much of this detail. However, because of the large area covered by fine detail, a great number of points are required to make up the difference. Therefore, a loose error threshold for point insertions is required to achieve acceptable performance and storage requirements.

Figure 2 shows renderings of several different representations of the chest data. The two hybrid renderings using a volumetric resolution of 128^3 in images (c) and (d), provide a large improvement over the 128^3 volume rendering alone in image (b). The $1/64$ error threshold in image (d) produces an improvement over the $1/32$ error threshold in image (c) by extending the very ends of the lung airways. The 256^3 volume rendering in image (e) is able to capture the larger airways, but notice that the color is slightly incorrect: the airways are more pink than those in the original. This is because the very low density areas around the airways smear out the

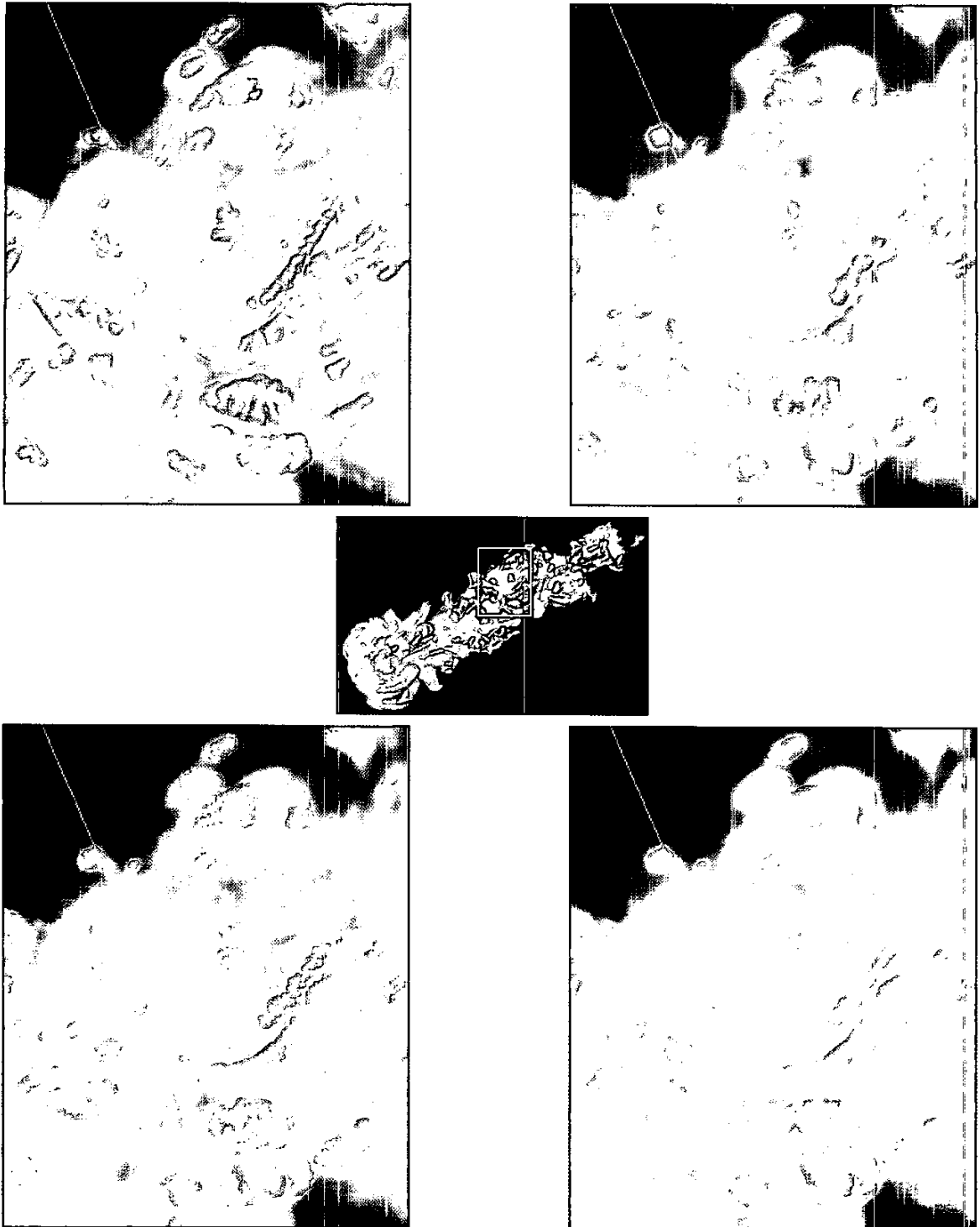


Figure 1: Close-up images comparing renderings of the argon bubble. The top row shows standard volume renderings of the original 512^3 volume (left) and the lower-resolution 256^3 volume (right). The bottom row shows the the same 256^3 volume with 361K points, rendered with square OpenGL points (left) and with Gaussian splats (right). The middle image is the entire view at 512^3 with the zoom area marked.

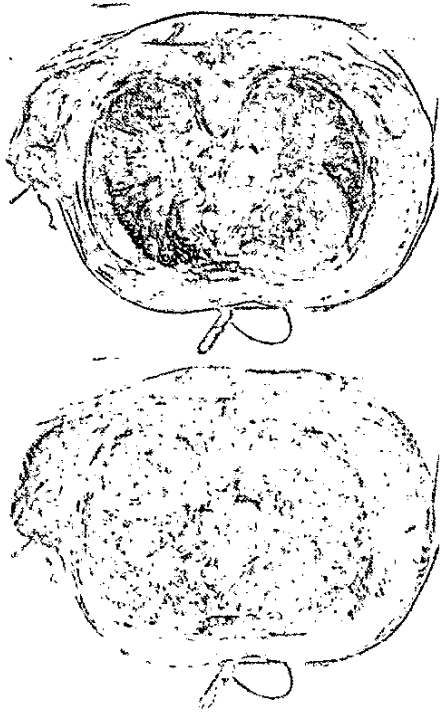


Figure 3: Pixel difference between a 512^3 resolution rendering and (top) a 128^3 volume rendering, and (bottom) the same 128^3 volume with 11 million points. Darker colors indicate regions of higher error.

higher values, resulting in density values that are too low. The hybrid representation in (f) is able to correct much of this error and add additional detail to the fine structures.

Figure 3 shows the improvement that the addition of points offer for a 128^3 volume representation. These images were generated by subtracting the standard and hybrid representations, respectively, from a 512^3 reference rendering, and inverting the result for better clarity in print. The hybrid rendering noticeably reduces the magnitude of the darker regions (those with highest error) of the difference image. The magnitude of the lightly-colored, low-error regions were not affected, because the error value of those regions was below the threshold set during point generation (in this case, $1/64$).

5.3 The Furby® Data Set

The hybrid rendering method is particularly well-suited for mechanical data. This type of data is often characterized by high resolution, sharp edges, fine detail, and large differences in density. The Furby data set is a $512 \times 512 \times 2048$ CT scan of a Furby, a mechanical toy that interacts with people by moving its body and playing recorded pieces of conversation. It consists of a fur-covered plastic shell (the fur is not resolved in the limited dynamic range of the data) containing a variety of electronic and mechanical parts. The original data was reconstructed from 361 X-ray views at a resolution of 1746×1869 .

The Furby exhibits many of the properties of mechanical data: it is too big to visualize in real time on commodity hardware (1GB with one-byte normals), density values are concentrated at the upper and lower extremes, and there are many small, sharp features. In the left column of Figure 4, the transfer function was selected to

provide a faint outline of the plastic body in red, and to highlight the high-density metal parts in yellow. The right column shows a slightly different transfer function and view angle, with the outline of the cloth exterior (and, unfortunately, the box used to hold the toy) in purple, and the metal parts in yellow and green.

Few metal parts are visible in the volume rendering (parts (a) and (b) of Figure 4) because the downsampling blurred these fine details into the surrounding low-density area (air). These areas have incorrect density values and so do not appear when the transfer function is selected to highlight metal parts. The larger metal parts that do appear are poorly resolved. However, in the hybrid renderings, the printed circuit board, with the electronics parts coming off the top, is clearly visible in the abdomen of the Furby. The hybrid method also restores the sharp corners on the metal bars used to pivot the joints, resolves the heads on the screws, and does a particularly good job on the wires leading to the sensors in the forehead.

Raising the resolution of the error sampling grid makes a visible difference in the quality of rendering generated, but requires the error threshold be raised to prevent too many points from being generated. Figure 4 shows the difference between error grids of 512^3 and 1024^3 . (Unfortunately, our preprocessing program does not yet handle non-cubic error grids, which prevent us from sampling at the original resolution of $512 \times 512 \times 2048$. Sampling at the original resolution should give slightly better results.)

Raising the error threshold for the 1024^3 error calculation grid above $1/8$ (the bottom row of Figure 4) to $1/12$ (not shown) makes almost no visible difference in the quality of rendering. This is because both $1/8$ and $1/12$ are enough to capture enough error from the very dense metal parts, and neither $1/8$ nor $1/12$ is small enough to capture error present in the low-density plastic and cloth regions. Raising the error threshold even further to $1/16$ starts to capture error for the plastic regions, but using a sampling grid of 1024^3 produces over 20 million points: too many to render in real time.

6 CONCLUSIONS

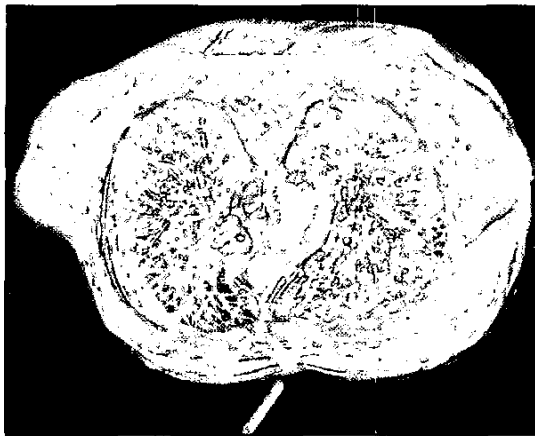
The hybrid point/volumetric storage and rendering method improves rendering quality available for a given amount of graphics hardware, and significantly reduces the amount of storage space required. It is able to work on many different types of volumetric data, including medical, mechanical, and simulation data.

While the hybrid method can effectively compress relatively small (less than 512^3) volumetric data sets with limited loss in detail, it is most useful for cases where the original data is too big to be rendered on a given computer. In these cases it allows interaction that was not previously possible, while preserving many details at their original resolution.

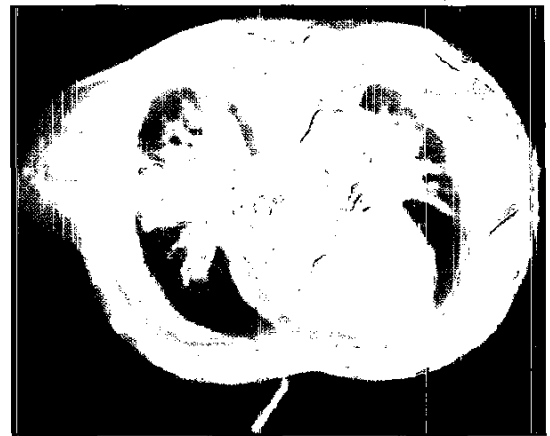
Our system does not yet handle error computation grids that are not cubes, and many scanning technologies can produce different resolutions for different axes. Changing the preprocessing program to handle these cases is straightforward. During viewing, however, the point size would need to be adjusted depending on the viewing angle to properly cover the area. We would also like to look at compressing time-varying data with a similar hybrid method, and using variable point sizes to allow further optimization.

7 ACKNOWLEDGMENTS

This work has been sponsored by the Los Alamos National Laboratory, DOE SciDAC, the National Science Foundation under contract ACI 9983641 (PECASE Award), and through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251. The authors are particularly grateful to those people who provided the test data sets. The shock-bubble



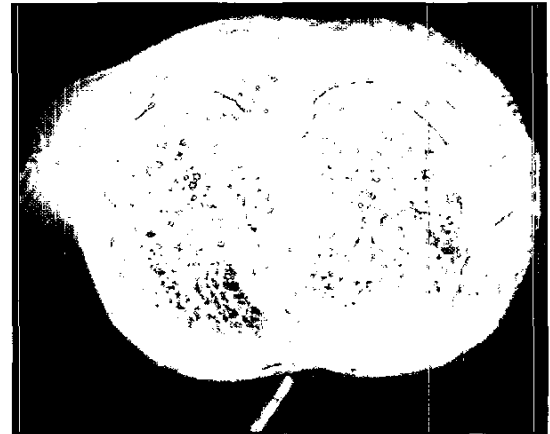
(a) 512^3 volume rendering: 0.35 s/frame, 268MB.



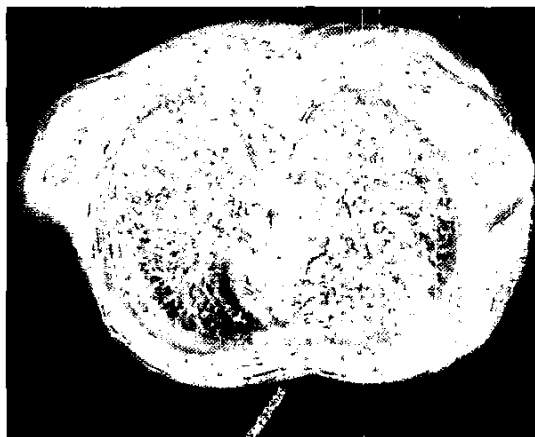
(b) 128^3 volume rendering: 0.01 s/frame, 4MB



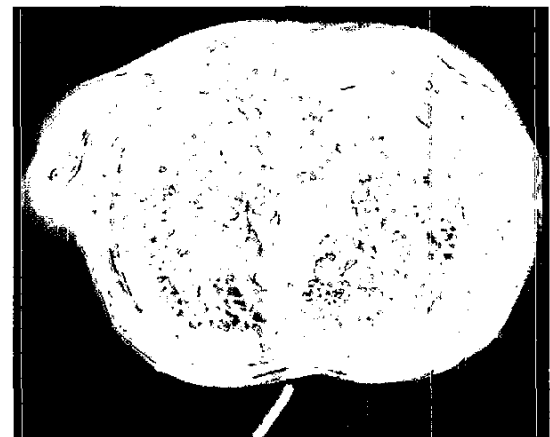
(c) 128^3 volume rendering with 3979K points: 512^3 error sampling grid, 1/12 error threshold, 0.29 s/frame, 40MB.



(d) 128^3 volume rendering with 10999K points: 512^3 error sampling grid, 1/64 error threshold, 0.75 s/frame, 103MB.

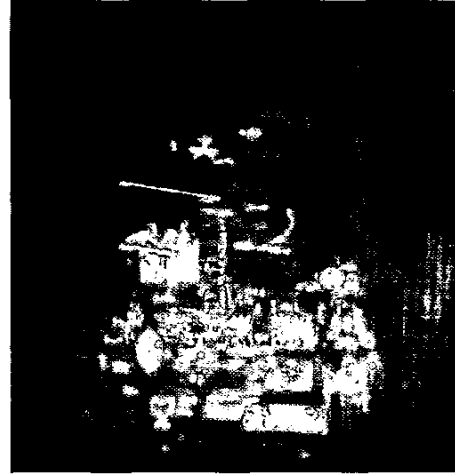
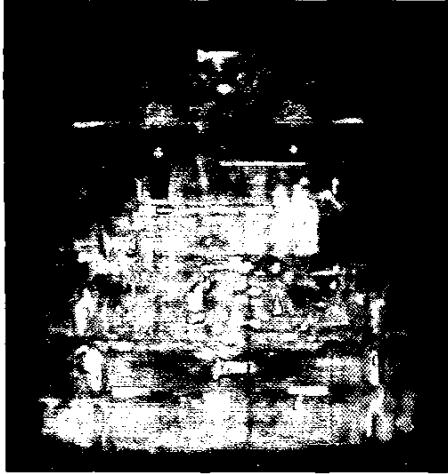


(e) 256^3 volume rendering: 0.04 s/frame, 33MB.

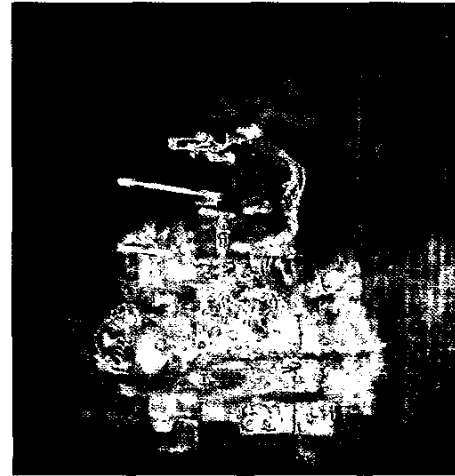
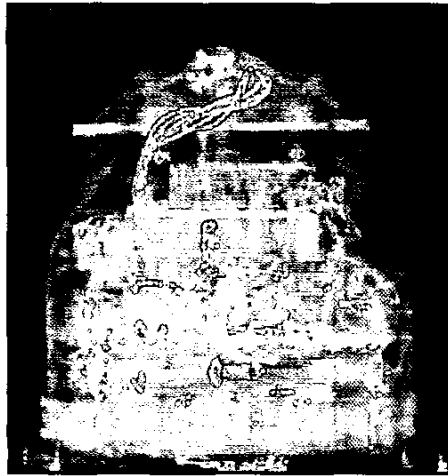


(f) 256^3 volume rendering with 7058K points: 512^3 error sampling grid, 1/64 error threshold, 0.47 s/frame, 97MB.

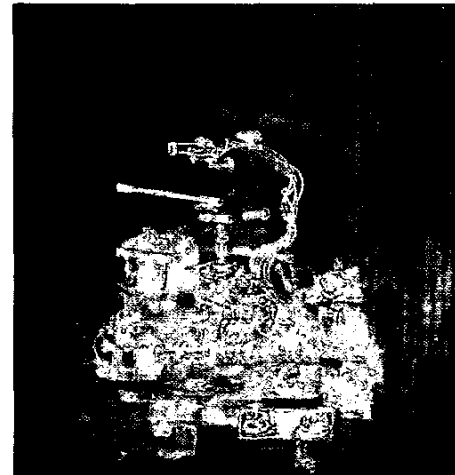
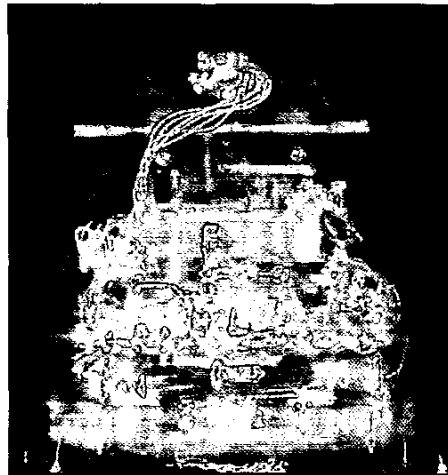
Figure 2: Comparison of rendering techniques of the chest MRI dataset. In all pictures, the front and back have been clipped by 20% to allow the detail of the lungs to be visible. Although rendering times for hybrid representations are not necessarily less than for the full-resolution data (depending on the number of points generated), they place much less stress on the memory systems, making it possible to display data that will not fit on the computer or video card at full-resolution.



(a) and (b): 256^3 volume rendering, 0.07 s/frame, 33MB.



(c) and (d): 256^3 volume rendering with 2.9M points, 512^3 error sampling grid, 1/16 error threshold, 0.36 s/frame, 59MB.



(e) and (f): 256^3 volume rendering with 4.7M points, 1024^3 error sampling grid, 1/8 error threshold, 0.58 s/frame, 71MB.

Figure 4: Comparison of hybrid rendering for the Furby data set. The original resolution was $512 \times 512 \times 2048$ (1GB with 1-byte normals), too big to display on commodity hardware. The hybrid technique is able to restore much of the detail lost by subsampling. The left column shows the Furby from the back, its plastic shell is visible in red, and the metal is visible in yellow. The right column shows the Furby at an angle, and reveals the faint purple outline of the fabric and the box used to hold it.

flow data set was provided by the Center for Computational Sciences and Engineering at the Lawrence Berkeley National Laboratory (<http://seesar.lbl.gov/ccse>). The MRI chest data set was available through Dr. H. Miyachi at Kubota Co., Japan. The Furby data set was provided by Anthony Davis at Hytec Inc. and Bill Ward of Los Alamos National Laboratory.

REFERENCES

- [1] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings Volume Visualization Workshop*, pages 91–98. ACM SIGGRAPH, October 1994. ISBN 0-89791-741-3.
- [2] B. Chamberlain, T. DeRose, D. Lischinski, D. Salesin, and J. Snyder. Fast rendering of complex environments using a spatial hierarchy. In *Graphics Interface '96*, pages 132–141. Canadian Information Processing Society / Canadian Human-Computer Communications Society, May 1996. ISBN 0-9695338-5-3.
- [3] H. E. Cline, W. E. Lorensen, S. Ludke, C. R. Crawford, and B. C. Teeter. Two algorithms for the three-dimensional construction of tomograms. *Medical Physics*, 52(15):320–327, 1988.
- [4] J. Grossman and W. Dally. Point sample rendering. In *Proceedings Eurographics Rendering Workshop 1998*, pages 181–192, 1998.
- [5] J. Kniss, P. McCormick, A. McPherson, J. Ahrens, J. Painter, A. Keahey, and C. Hansen. Interactive texture-based volume rendering for large data sets. *IEEE Computer Graphics and Applications*, 21(4):52–61, 2001.
- [6] E. LaMar, B. Hamann, and K. Joy. Multiresolution techniques for interactive texture-based volume visualization. In *Proceedings Visualization 99*, pages 355–361. ACM Press, New York, 1999.
- [7] D. Laur and P. Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. In *Computer Graphics (Proceedings of SIGGRAPH 91)*, volume 25, pages 285–288, Las Vegas, Nevada, July 1991. ISBN 0-201-56291-X.
- [8] M. Levoy and T. Whitted. The use of points as a display primitive. Technical Report TR- 85-022, The University of North Carolina at Chapel Hill, 1985.
- [9] E. Lum and K.-L. Ma. Hardware-accelerated parallel non-photorealistic volume rendering. In *Proceedings of International Symposium on Nonphotorealistic Rendering and Animation*, June 2002.
- [10] K.-L. Ma, G. Schussman, B. Wilson, K. Ko, J. Qiang, and R. Ryne. Advanced visualization technology for terascale particle accelerator simulations. In *Proceedings of Supercomputing 2002 Conference*, November 2002.
- [11] X. Mao. Splatting of non-rectilinear volumes through stochastic resampling. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):156–170, June 1996.
- [12] J. Meredith and K.-L. Ma. Multiresolution view-dependent splat based volume rendering of large irregular data. In *Proceedings Parallel and Large Data Visualization Symposium*, pages 93–99, October 2001.
- [13] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. The VolumePro real-time ray-casting system. In Alyn Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, pages 251–260, Los Angeles, 1999. Addison Wesley Longman.
- [14] J. Qiang and R. Ryne. Beam halo studies using a 3-dimensional particle-core model. *Physical Review Special Topics - Accelerators and Beams*, 3(064201), 2000.
- [15] J. Qiang, R. Ryne, S. Habib, and V. Decyk. An object-oriented parallel particle-in-cell code for beam dynamics simulation in linear accelerators. *Journal of Computational Physics*, 163(434), 2000.
- [16] W. T. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics*, 2(2):91–108, April 1983.
- [17] W. T. Reeves and R. Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. In *Computer Graphics (Proceedings of SIGGRAPH 85)*, volume 19, pages 313–322, San Francisco, California, July 1985.
- [18] S. Rusinkiewicz and M. Levoy. Qsplat: A multiresolution point rendering system for large meshes. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 343–352. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, July 2000. ISBN 1-58113-208-5.
- [19] E. Swan II, K. Mueller, T. Möller, N. Shareef, R. A. Crawford, and R. Yagel. An anti-aliasing technique for splatting. In *IEEE Visualization '97*, pages 197–204. IEEE, November 1997. ISBN 0-58113-011-2.
- [20] M. Wand, M. Fischer, I. Peter, F. Meyer auf der Heide, and W. Straßer. The randomized z-buffer algorithm: Interactive rendering of highly complex scenes. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 361–370. ACM Press / ACM SIGGRAPH, August 2001. ISBN 1-58113-292-1.
- [21] M. Weiler, R. Westermann, C. Hansen, K. Zimmerman, and T. Erl. Level-of-detail volume rendering via 3d textures. In *Proceedings of IEEE Volume Visualization 2000*, 2000.
- [22] L. Westover. Interactive volume rendering. In *Proceedings Volume Visualization Workshop*, pages 9–16. The University of North Carolina at Chapel Hill, 1989.
- [23] O. Wilson, A. Van Gelder, and J. Wilhelms. Direct volume rendering via 3d textures. Technical Report UCSC-CRL-94-19, Jack Baskin School of Eng., Univ of California at Santa Cruz, 1994.
- [24] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. EWA volume splatting. In *Proceedings Visualization 2001*, pages 29–36. ACM Press, New York, October 2001.