

# VISUALIZING TIME-VARYING VOLUME DATA

*This article reviews strategies developed so far for enabling interactive visualization of volume data from time-varying simulations with a focus on encoding, feature extraction, and rendering issues. The author also discusses emerging trends in time-varying data visualization research and their potential impact on the scientific research community.*

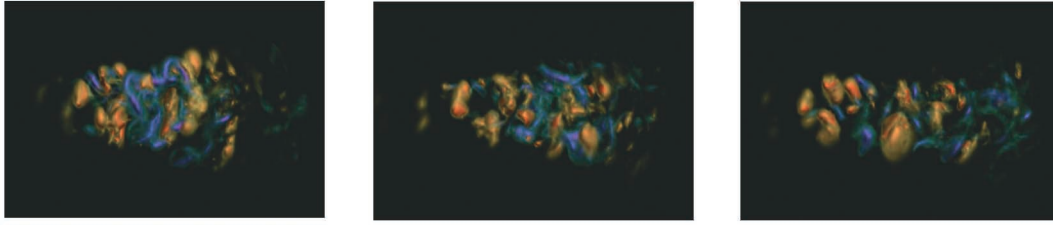
Studying dynamic aspects of physical and chemical processes is critical to the advances of many sciences. State-of-the-art scientific computing technologies allow accurate numerical modeling of many physical and chemical processes in their spatial and temporal domains. However, an increasingly challenging problem scientists must face is how to effectively explore and understand the resulting time-varying volume data that is large in space (from 100 million to a billion voxels), time (hundreds to thousands of time steps), and variable domain (from five to over 100 variables).

Scientists' ability to visualize time-varying phenomena is absolutely essential to ensure correct interpretation and analysis, provoke insights, and communicate those insights to others. For instance, by appropriately rendering a time-varying data set, scientists can produce an animation sequence that illustrates how selected underlying structures in the data evolve over time. In particular, interactive visualization lets scientists freely explore the spatial and temporal

domains in both the variable space and visualization parameter space. Figure 1 presents selected time steps from the visualization of a turbulent jet data set. The figure demonstrates scientists' ability to interactively explore the data and derive visualizations that clearly separate vortices with positive and negative values. To achieve the needed interactivity, a visualization solution's design must account for hardware constraints. How fast can a sequence of time steps be transferred to the rendering engine, and how can we accelerate the rendering calculations?

The availability of texture hardware support for volume rendering enables real-time volume visualization.<sup>1</sup> However, the sheer size of a data set from a contemporary scientific application can easily overwhelm the texture memory space of a typical graphics-acceleration card designed for video games. How to reduce a data set's storage requirement without removing fine features in the data is thus central to time-varying data visualization research. Should we do data reduction at the simulation time or as a postprocessing task? Is lossy compression acceptable? Is automatic feature extraction possible? Is there a way to couple these data-reduction strategies with rendering?

This article reviews the strategies developed so far for visualizing volume data from time-varying simulations. My work, and therefore this



**Figure 1.** Selected time steps from the visualization of a turbulent jet data set. Positive-value vortices are green and blue, while negative vortices are yellow and red.

article's focus, concentrates on encoding, feature extraction, and rendering issues. I also discuss emerging trends in time-varying data visualization research and their potential impacts on the scientific research community.

### Encoding

We can reduce a volume data set's size and therefore make it more manageable by using compression. The advantages of compressing volumetric data are twofold. First, it reduces the data's storage requirements. This could let a data set fit in texture memory that might otherwise not fit, eliminating the need for transferring data across the graphics bus from main memory to texture memory. It is also possible to use storage reduction to fit relatively large compressed data sets entirely in main memory, thus eliminating the need for swapping from disk. The other benefit of compression is the reduction in I/O. Even if a data set does not fit into texture memory, transferring compressed data across the graphics bus can be substantially faster than with uncompressed data, allowing interactive visualization. In addition, for out-of-core rendering of very large data sets, reading compressed data from disk requires less time than reading uncompressed data.

There are two basic approaches to compressing time-varying volume data. The first approach is to separate the time dimension from the spatial dimensions. For example, differences in encoding the data coherence between consecutive time steps can result in a significant reduction.<sup>2</sup> However, this approach is limited to sequential browsing of the data's temporal aspect. That is, browsing must start from the first time step. We can use quantization together with octree and difference encoding to effectively compress both the spatial domain and temporal domains of the data.<sup>3</sup> Subtrees can be merged for consecutive time steps. During rendering, for stationary view, we can reuse partial images built from subtrees that do not change in the subsequent time steps.

It is often desirable to have an underlying analysis model for characterizing time-varying data. We can accomplish such a model by wavelet encoding of each time step separately to derive compressed multiscale tree structures.<sup>4</sup> By examining the resulting multiscale tree structures and wavelet coefficients, we can perform feature extraction, tracking, and further compression. It is also possible to compress time-varying isosurfaces and associated volumetric features with wavelet transform to allow fast reconstruction and rendering.<sup>5</sup>

The second approach is to treat time-varying volume data as 4D data. For example, the 4D data can be encoded with a 4D tree (an extension of octree<sup>6</sup>) and the associated error/importance model to control compression rate and image quality. A more refined design is based on a 4th-root-of-2 subdivision scheme coupled with a linear B-spline wavelet scheme for representing time-varying volume data at multiple levels of detail.<sup>7</sup>

Whether we should treat time-varying volume data as 4D data or not should depend on the characteristics of the data. For example, if the discrepancy between the temporal and spatial resolutions would be large, it could become difficult to locate the temporal coherence in certain subdomains of the data. Consequently, we should consider the time domain separately for encoding. Another problem with using 4D trees is that coupling spatial and temporal domains makes it difficult to locate regions with only temporal coherence but not spatial coherence.

A time-space partitioning (TSP) tree<sup>8</sup> is a hierarchical data structure for a better use of spatial and temporal coherence. In essence, a TSP tree's skeleton is a standard complete octree, which recursively subdivides the volume spatially until all subvolumes reach a predefined minimum size. To store the temporal information, each TSP tree node is a binary tree. Every node in the binary time tree represents a difference time span for the same subvolume in the spatial domain. The objective of this design is to reduce the amount of data required to complete the

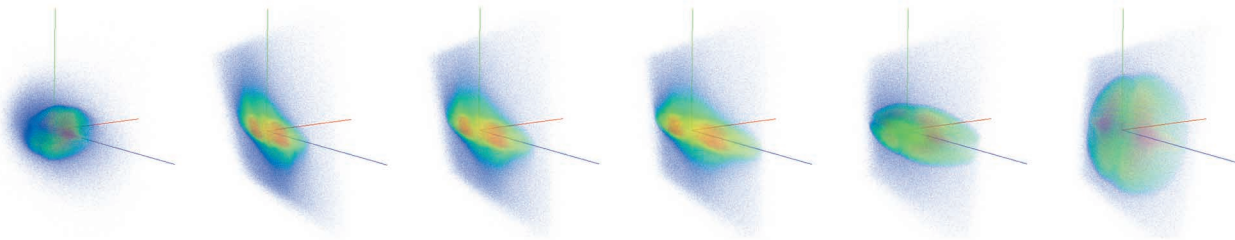


Figure 2. Selected time steps from the visualization of a particle beam data set using a hybrid rendering technique.<sup>11</sup> (Images generated by Brett Wilson.)

rendering task and to reduce the volume rendering time. In particular, TSP trees let the renderer use data from subvolumes of different spatial and temporal resolutions.

### Feature Extraction

Visually scanning through terabytes of data for features, if not impossible, is a labor-intensive task. Automated feature extraction using domain knowledge can significantly cut down the storage requirements and rendering cost of visualization tasks. For example, it is possible to track vortices in an unsteady flow as they appear, branch, merge, and disappear over time. Previous efforts<sup>9,10</sup> show several orders of magnitude savings in storage requirements, which enables interactive browsing through the extracted features on an average graphics workstation.

Another example is visualizing particle-beam data obtained from particle-accelerator simulations. The data set typically consists of hundreds of millions to billions of particles for each time step, making it impossible to render in real time on a desktop PC. One solution is to use a coarse volume representation for the overall beam and a point-based representation for selected features of interest, one of which in this case is the outer layer of the particle beam called a *halo*. An image is generated by mixing hardware-accelerated rendering of the overall beam's volumetric representation with a point-based rendering of the halo region. The overall savings in storage requirements makes it possible to load numerous time steps into the main memory for interactive visualization. Scientists can then examine the behavior of individual particles in the halo region. Figure 2 shows selected frames from an animation of a disoriented beam.

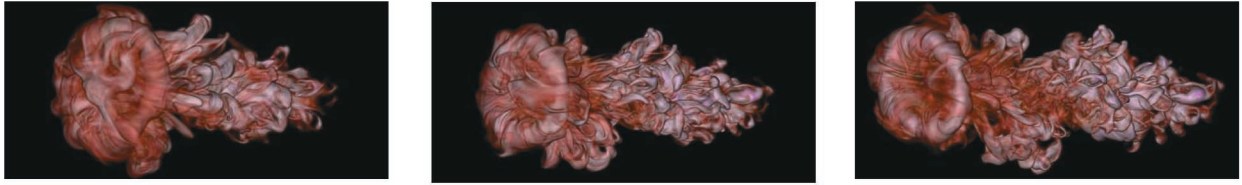
If it is possible to reliably capture features at different scales that completely characterize the data set, we can then integrate this feature ex-

traction process into the simulation process so that the simulation's outputs become the extracted features rather than arrays of numbers. In this way, the overall scientific discovery process is optimized by reducing both the data transfer and storage costs early in the data-analysis pipeline. This is the optimal goal of feature extraction. In practice, scientists have not adopted this approach for two reasons. First, most scientists are reluctant to use their supercomputer time for visualization calculations. Second, they must understand the features well enough to make the automated extraction effective. The common practice is storing only selected time steps of the data or studying the stored data at a coarser resolution, which defeats the original purpose of performing the high-resolution simulations.

### Rendering

Although appropriate encoding can facilitate the subsequent rendering step, several challenging issues associated with rendering large time-varying data still exist. When the available computing resources cannot accommodate the data set for scientists to directly interact with the data and freely browse through it in both the data's spatial and temporal domains, we must seek alternative representations of the data or additional computing resources. The former has to do with additional compression, automated transfer function generation, or feature extraction. The latter involves using parallel distributed computing.

Selection of color and opacity transfer functions determines how the information in the volume data is presented visually. If we can automatically generate the transfer functions that effectively capture the data's essence, it is possible to produce an animation of the data in a batch-mode step. On the other hand, rendering



**Figure 3. Selected time steps from a visualization of an argon shock-bubble data set. The transfer function I used was automatically generated.<sup>13</sup> (Images generated by T.J. Jankun-Kelly.)**

extracted features (such as vortices, shocks, or streamsurfaces) rather than original volume data can significantly cut down the storage requirements and rendering cost, leading to more efficient data exploration.

### Transfer Functions

Batch-mode rendering requires preselected transfer functions. Although researchers have extensively studied the problem of defining transfer functions for a single volume data,<sup>12</sup> properly using and creating transfer functions for time-varying volume data is still unaddressed.

In time-varying volume visualization, the phenomena under study evolves in some manner over time and space. Features of interest in a time series might exhibit a regular, periodic, or random pattern. A regular pattern is characterized by a feature that moves steadily through the volume. The feature's structure neither varies dramatically nor follows a periodic path. Features exhibiting a periodic pattern appear and disappear over time. Transient features of interest or features that fluctuate randomly are common, such as those found in turbulent flows. Generally, we can more easily detect and more efficiently render regular and periodic patterns.

Therefore, transfer functions for time-varying data need to capture one of the aforementioned three behaviors. Ideally, we want to capture these behaviors by using a single transfer function. In practice, this is not always possible, especially when several different types of features exist in the data and persist for different lengths of time. Using more than one opacity transfer function must be done with care because the transition between two transfer functions can result in a sudden change in the visualization that is physically misleading.

There are several different ways to generate a single or minimal set of transfer functions for rendering time-varying volume data.<sup>13</sup> Volumes possessing features with a regular structure, regardless of their motion, often can be adequately

rendered with a single transfer function. For data sets with dynamic boundaries, multiple transfer functions are needed to capture essential features appearing in each specific time span. By measuring the coherence level of the data in the time dimension, it is possible to identify individual time spans for each of which a transfer function can be defined in a straightforward manner. Figure 3 displays three selected time steps of an animation produced by using an automatically generated transfer function. The animation succinctly captures the impact of the shock over an argon bubble.

### Hardware-Accelerated Rendering

Hardware-accelerated volume rendering requires that we load the volume data into the video card's texture memory prior to rendering. The volume size that we can render interactively is thus limited by the amount of video memory the card contains. This is because the access and transfer of data from main memory across the graphics bus is relatively slow compared to the direct access of graphics memory. Time-varying hardware-accelerated volume rendering exacerbates this problem by also requiring transferring data from disk to main memory. Joe Kniss and his colleagues demonstrate interactive texture-based volume rendering of large time-varying data using a 16-pipe SGI Origin 2000, but the overall performance was limited by how fast each time step can be brought into the texture memory from the disk.<sup>14</sup> David Ellsworth, Ling-Jen Chiang, and Han-Wei Shen develop a rendering method that better utilizes 3D texture memory by exploiting a TSP tree representation of the time-varying volume data.<sup>15</sup>

One solution for this I/O bottleneck is to treat video memory, main memory, and disk as a three-level cache for volume rendering. By compressing the volume data, we increase the amount of data that can fit in each level while decreasing the I/O costs of transferring data between these levels. In this way, the interactive

volume rendering of large data sets is possible using commodity PC hardware.

I along with Eric Lum and John Clyne<sup>16</sup> introduce a low-cost strategy for visualizing time-varying volume data that integrates lossy compression and rendering to take full advantage of the increasing power and decreasing cost of commodity PC graphics cards. The basic idea is based on the temporal encoding of indexed volumetric data that can quickly be decoded in hardware. The actual implementation extensively uses OpenGL's support for changing color palettes without reloading volume textures. The cycling of color palettes can be used to create simple animations from static images. Similarly, using color palette manipulation lets a single scalar value represent multiple time steps' values. Even though this results in lossy compression, according to our test results,<sup>16</sup> in most cases, the differences between the compressed and uncompressed visualizations are hard to discern.

Using four times compression on an AMD 1.2-GHz Athlon with 768 Mbytes of main memory and a GeForce 3 with 64 Mbytes of texture memory, we can render 1,492 time steps of a  $256 \times 256 \times 256$  volumetric data set in an out-of-core fashion at approximately 6.8 frames per second using 256 object-aligned textured polygons. Because the main memory can hold 280 time steps of the data, if rendered in-core, we can achieve 25.8 frames per second. Without compression, the same 280 time steps no longer fit in main memory and would need to be swapped into main memory in an out-of-core manner. A memory-resident subset of this uncompressed data can be rendered at about 11.5 frames per second, compared to the 25.8 frames per second with compression. We obtained these results when rendering the volume to a  $512 \times 512$  window, with the volume occupying more than half the window area. Using a cluster of eight PCs, we can render time-varying  $512 \times 512 \times 512$  volume data sets in the same fashion at interactive rates.<sup>17</sup>

There is a distinct trade-off between the compression ratio and rendering performance versus the compressed volume's quality. This gives users a degree of flexibility in choosing the compression ratios that best meet their needs. For example, if scientists are interested in viewing a short time sequence at high quality, they can use the lower compression ratio. To view a long sequence of data at high speeds, they can select a higher compression rate. Scientists can combine compression ratios to preview a data set at a coarser temporal resolution and then view a spe-

cific time sequence of interest with less compression. This hardware-accelerated decoding and rendering approach clearly shows the feasibility of putting a PC-based system on every scientist's desktop, making interactive exploration of large data sets accessible to a far broader group of scientists and engineers.

### Parallel Rendering

When the volume data is too large for a single computer to render interactively, or the display resolution requirement is high (as for a display wall), an obvious solution is to use parallel rendering by distributing both the data and rendering calculations. Researchers have introduced several parallel volume-rendering algorithms,<sup>18-20</sup> but they are not for efficiently rendering time-varying data. The time needed to load each time step of the data set into the main memory alone can inhibit interactive rendering.

A scientist who routinely performs large-scale simulations on a parallel computer operated at a supercomputer center typically leaves the large output data on the center's mass storage device. To visualize and study the data, scientists remotely access the same parallel computer or a different one to perform the needed visualization calculations. The resulting images or animations are delivered to the scientist's desktop computer for viewing. This postprocessing of precalculated data remains a common practice.

Without a high-speed network and parallel I/O support, two bottlenecks make interactive visualization impossible for such a postprocessing scenario. One is the need to read large files continuously or periodically throughout the course of the visualization process. The other is the delay due to transferring the resulting images over a nondedicated network.

One way to remove the bottlenecks is to employ parallel pipelined rendering.<sup>21</sup> Along with careful grouping of processors, pipelining can not only hide I/O overheads but also maximize processor utilization. Furthermore, visually lossless compression can significantly cut down the cost of transferring output images to a display device through a WAN. With this remote visualization strategy, David Camp and I have shown that the display rates on a desktop PC can stay close to the interactive rendering rates of a 256-node PC cluster located over 6,200 miles away.<sup>21</sup>

Using such a parallel-pipelined renderer, the user can choose to minimize either the interframe delay or the overall rendering time. The former ensures a particular level of interactivity

whereas the latter cuts down as much as possible the turnaround time for batch-model rendering.

To achieve highly efficient visualization of large data, we must carefully optimize every step in the visualization pipeline. Parallelizing the rendering step alone is likely insufficient. The parallel pipelined approach can address the I/O bottlenecks. Another critical bottleneck is concerned with the final image composition step required by the widely used sort-last parallel rendering. Especially when the rendering calculations are hardware accelerated, this compositing step could severely limit the scalability of the overall rendering performance. Fortunately, inexpensive, out-board compositing hardware has become more widely available so we can build a graphic-enhanced PC cluster to deliver scalable rendering performance.

### Remaining Challenges

Research so far in time-varying volume data visualization has primarily addressed the problems of encoding and rendering a single scalar variable

on a regular grid. There is also the need to simultaneously visualize multiple variables. Encoding a time-varying vector field has largely been unexplored. Time-varying unstructured-grid data sets has been either rendered in a brute-force fashion or just resampled and downsampled onto a regular grid for further visualization calculations. An even more challenging problem is visualizing data on a mesh structure that changes over time.

### Multiple Variables

In practice, almost all data sets obtained from numerical modeling of physical phenomena or chemical process record multiple scalar and vector properties at each mesh point. In medicine, increasingly there is also the need to study time-varying volumetric data from several different imaging modalities.<sup>22</sup> The need to visualize vector quantities or multiple variables simultaneously would require refined designs of the encoding and rendering processes. Although each variable generally should be encoded independent of other variables, in some situations, it

## Career Opportunities

### UNIVERSITY OF UTAH Tenure-Track Faculty Position Scientific Computing and Imaging Institute and Bioengineering Department

Applications are invited for an assistant professor level, tenure-track faculty position at the Scientific Computing and Imaging (SCI) Institute and the Department of Bioengineering at the University of Utah. The SCI Institute is an interdisciplinary research institute consisting of approximately 65 scientists, staff, and students dedicated to advancing the development and application of computing, scientific visualization, and numerical mathematics to topics a wide variety of fields such as bioelectric fields in the heart and brain, multimodal imaging, and combustion. The SCI Institute currently houses two National research centers: the NIH Center for Bioelectric Field Modeling, Simulation, and Visualization and the DOE Advanced Visualization Technology

Center.

The Bioengineering Department has an international reputation for research and graduate education with particular strengths in biobased engineering, biomaterials, biomechanics, biomedical computing/imaging, controlled chemical delivery, tissue engineering and neural interfaces. Tenure-track faculty typically has primary appointments within College of Engineering and secondary appointments within the Health Sciences. The Department is home to approximately 100 graduate students and 90 upper-level undergraduate students.

The successful candidate will be expected to maintain/establish a strong extramurally funded research program consistent with the research mission of the SCI Institute, and participate in undergraduate/graduate teaching consistent with the educational mission of the Department of Bioengineering.

The candidate should have a doctoral degree in a field related to biomedicine or engineering and have demonstrated research skills, ideally with 2 or more years of postdoctoral experience. A strong record of experience in the application of computational techniques to one or more fields of biomedical research is also necessary. Specific areas of relevant, established

strength in the SCI Institute include cardiac and neurologic electrophysiology, biomedical image and signal processing, and bioelectric and biomagnetic fields. The candidate must be prepared to seek and secure ongoing extramural research support, collaborate closely with researchers in interdisciplinary projects, and establish or maintain an international presence in his or her field.

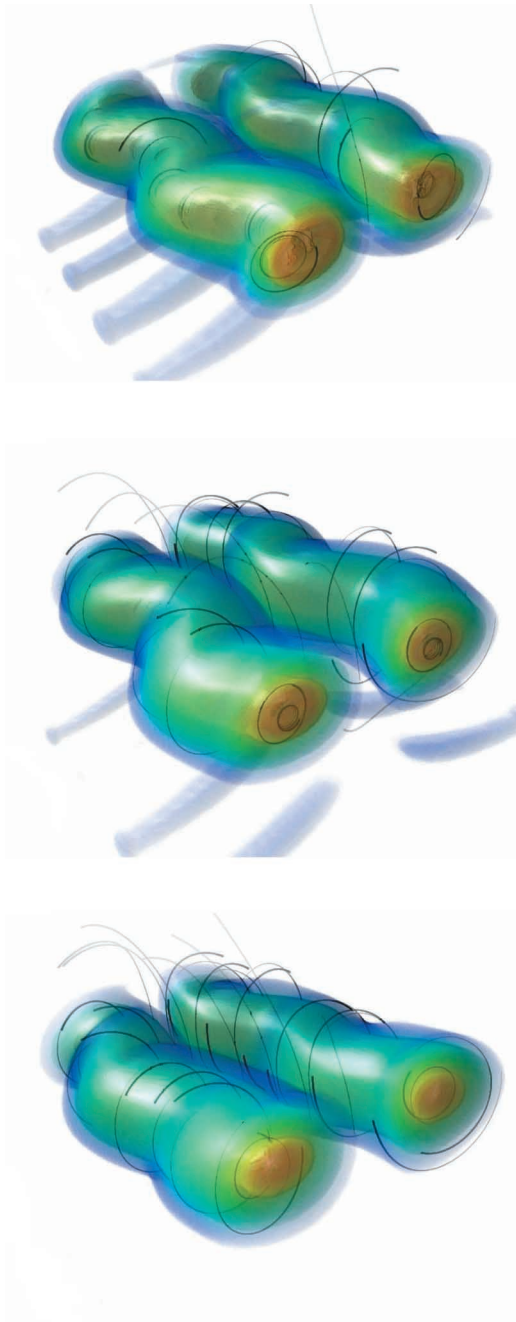
A complete CV, names of three references, and a short description of current research activities, teaching experience, and career goals should be sent to:

Director  
Scientific Computing and  
Imaging Institute,  
University of Utah  
50 So. Central Campus Drive, Rm. 3490  
Salt Lake City, UT 84112  
Email: crj@sci.utah.edu  
Web: www.sci.utah.edu

Review of applications will begin December 2002 and continue until selection of a candidate is complete.

The University of Utah is an Equal Opportunity, Affirmative Action employer and encourages nominations and applications from all qualified individuals including women and minorities and provides reasonable accommodation to the known disabilities of applicants and employees.

**Figure 4.** Simultaneous visualization of velocity and vorticity fields from the modeling of aircraft wake vortices.<sup>23</sup> From top to bottom, we see three selected time steps. Velocity direction is shown with stroke rendering, and the vorticity field is volume rendered. (Aleksander Stoppel generated these images.)



could be advantageous to compress one variable by taking into account the values of other variables. One example is to select a particular compression level for a scalar field according to the corresponding velocity field.

Extended rendering capabilities are necessary for visualizing multiple variables. The general approach is to use one or more properties of the data for the enhanced rendering of another property of the data. Numerous possible combinations exist that we can use to achieve vari-

ous types or levels of enhancement. We can use multidimensional transfer functions for the rendering of multivariate volume data.<sup>23</sup> The key, again, is to ensure interactive visualization so that the user can freely explore different combinations for specific feature enhancements.

The new generation of PC graphics cards can support different rendering styles at interactive rates for the visualization of multiple variables.<sup>24</sup> Generally, our goal is to either highlight important features in another variable or add contextual information to the visualization. For example, Figure 4 shows simultaneous visualization of vorticity and velocity fields. The vorticity field is volume rendered, and the velocity field is stroke rendered. The user can interactively increase the strokes density and vary the stroke rendering style.

Because the cost of computing velocity field lines on the fly can be prohibitively high, to achieve interactive visualization, we must pre-compute field lines and store them hierarchically so that visualization can be progressively refined.

Finally, it is desirable to enhance temporal features, which is possible by applying hardware-accelerated, temporal-domain filter to reinforce the perception of the changing features in the data over time. Figure 5 shows this type of enhancement. Fast-moving features are more opaque and red.

### Irregular Meshes

Increasingly, unstructured grid methods are used in large-scale scientific computing to model problems involving complex geometries. By applying finer meshes only to regions requiring high accuracy, we can reduce computing time and storage space. On the other hand, using unstructured grid discretizations complicates the visualization task because the resulting data sets are irregular geometrically and topologically. The need to store and access additional information about the structure of the grid can lead to visualization algorithms that incur considerable memory and computational overhead.

By simply disregarding the unstructured mesh, we could encode time-varying scalar data the same way for regular-grid data. The question is if we can achieve better compression ratios by taking into account the mesh structure information. How can we compress the mesh structure information along with the scalar or vector data to facilitate the subsequent visualization calculations? For mesh structure modified by the simulation over time, is temporal-domain compression still feasible?




**Figure 5. Selected time steps from the visualization of the rotation flow data set. The fast-moving features are more opaque and red.<sup>23</sup> (Aleksander Stompel generated these images.)**

Jeremy Meredith and I have developed a meshless approach for interactive previewing of large, unstructured grid data.<sup>25</sup> It is straightforward to incorporate a multiresolution framework such that high-quality visualization can still be made as needed in an incremental manner. It is also possible to extend this technique with temporal-space encoding for the visualization of time-varying data.

**T**imely development of visualization technologies for large time-varying data is clearly still necessary. Application scientists must work together with visualization researchers to develop technologies that best meet the application requirements with the available computing resources. In addition to the development of more effective physically based feature extraction methods and tightly coupled encoding and rendering methods, the most promising approach seems to be simulation-time processing for visualization.

For scientists who run large-scale time-dependent simulations on parallel supercomputers operated remotely, such as a national supercomputer center, moving a complete time-varying data set from the supercomputer center to the scientist's own computing laboratory for data analysis and visualization can be troublesome, if not impossible. A viable solution to this problem is based on a simulation-time visualization scenario. That is, visualizing time-varying data probably can be done most efficiently and economically while the data are being generated, so that users receive immediate feedback on the subject under study and so that the visualization results can be stored (rather than the much larger raw data). Several researchers have demonstrated simulation-time visualization of time-dependent simulations. One type operates with tightly coupled parallel simulation and visualization to enable runtime visualization.<sup>26,27</sup> The simulation

and rendering calculations are performed on the same parallel computer. The other type relies on a more loosely coupled setting. VISUAL3<sup>28</sup> and SCIRun<sup>29</sup> are among the few coherent problem-solving environments that can support runtime tracking in this manner. SCIRun moves one step further to allow computational steering.

Saving images rather than raw data is not always acceptable, especially if the important information in the data cannot be captured with a few simple rules. We need to improve our ability to perform automatic feature extraction and to define adequate transfer functions with minimum hints from the user. Simulation-time processing to compress and reorganize the raw data into a form facilitating interactive browsing ought to be an included feature of future simulation programs. The task of visualizing large time-varying data then becomes switching between using a low-cost desktop previewing method for defining key visualization parameters and running a batch-mode parallel high-fidelity renderer for an in-depth study of selected aspects of the data. Scientists who have not been able to study their simulation data fully at the highest possible resolution should rethink their current approach to the visualization and data-understanding problem. It is timely to adopt a new paradigm for data analysis and visualization that is more integrated into their simulations and to also take advantage of the emerging interactive visualization technology. 

### Acknowledgments

*This work has been sponsored in part by the National Science Foundation PECASE award, NSF Large Scientific and Software Data Set Visualization (LSSDSV) program, Department of Energy Scientific Discovery through Advanced Computing (SciDAC) program, Los Alamos National Laboratory, Lawrence Berkeley National Laboratory, and Lawrence Livermore National Laboratory.*

Robert Wilson at the University of Iowa provided the turbulent jet data set. The Center for Computational Sciences and Engineering at LBNL (<http://seesar.lbl.gov/ccse>) provided the argon shock-bubble data set. Robert Ryne and Ji Qiang at the LBNL provided particle-beam data set. Charlie Zheng at Kansas State University made the wake vortices data set available. Finally, Pierre Lallemand at ASCI of CNRS in France provided the rotation flow data set. T.J. Jankun-Kelly, Eric Lum, Aleksander Stompel, and Brett Wilson provided the images in this article. I am grateful to all these people.

## References

- B. Cabral, N. Cam, and J. Foran, "Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware," *Proc. 1994 Symp. Volume Visualization*, IEEE CS Press, 1994, pp. 91–98.
- H.-W. Shen and C.R. Johnson, "Differential Volume Rendering: A Fast Volume Visualization Technique for Flow Animation," *Proc. Visualization '94 Conf.*, IEEE CS Press, 1994, pp. 180–187.
- K.-L. Ma and H.-W. Shen, "Compression and Accelerated Rendering of Time-Varying Volume Data," *Proc. 2000 Int'l Computer Symp. - Workshop on Computer Graphics and Virtual Reality*, 2000, pp. 82–89.
- R. Westermann, "Compression Time Rendering of Time-Resolved Volume Data," *Proc. Visualization '95 Conf.*, IEEE CS Press, 1995, pp. 168–174.
- B.-S. Sohn, C. Bajaj, and V. Siddavanahalli, "Feature Based Volumetric Video Compression for Interactive Playback," *Proc. Volume Visualization and Graphics Symp. 2002*, IEEE CS Press, 2002, pp. 89–96.
- J. Wilhelms and A. Van Gelder, "Multidimensional Trees for Controlled Volume Rendering and Compression," *Proc. 1994 Symp. Volume Visualization*, IEEE CS Press, 1994, pp. 27–34.
- L. Linsen et al., "Hierarchical Representation of Time-Varying Volume Data with '4th-root-of-2' Subdivision and Quadrilinear B-Spline Wavelets," *Proc. 10th Pacific Conf. Computer Graphics and Applications - Pacific Graphics 2002*, IEEE CS Press, 2002, pp. 346–355.
- H.-W. Shen, L.-J. Chiang, K.-L. Ma, "A Fast Volume Rendering Algorithm for Time-Varying Field Using A Time-Space Partitioning (TSP) Tree," *Proc. Visualization '99*, IEEE CS Press, Calif., 1999, pp. 371–377.
- R. Samtaney et al., "Visualizing Features and Tracking their Evolution," *Computer*, vol. 27, no. 7, 1994, pp. 20–27.
- D. Banks and B. Singer, "A Predictor-Corrector Technique for Visualizing Unsteady Flow," *IEEE Trans. Visualization and Computer Graphics*, vol. 1, no. 2, 1995, pp. 151–163.
- K.-L. Ma et al., "Advanced Visualization Technology for Terascale Particle Accelerator Simulations," *Proc. Supercomputing 2002 Conf.*, CD-ROM, IEEE CS Press, 2002.
- P. Hanspeter et al., "The Transfer Function Bake-Off," *IEEE Computer Graphics & Applications*, vol. 21, no. 3, 2001, pp. 16–22.
- T. Jankun-Kelly and K.-L. Ma, "A Study of Transfer Function Generation for Time-Varying Volume Data," *Proc. Volume Graphics 2001 Workshop*, Springer, pp. 51–65.
- Joe Kniss et al., "Interactive Texture-Based Volume Rendering for Large Data Sets," *IEEE Computer Graphics & Applications*, July/Aug. 2001, pp. 52–61.
- D. Ellsworth, L. Chiang, and H.-W. Shen, "Accelerating Time-Varying Hardware Volume Rendering Using TSP Trees and Color-Based Error Metrics," *Proc. 2000 Symp. Volume Visualization*, ACM Press, 2000, pp. 119–128.
- E. Lum, K.-L. Ma, and J. Clyne, "Texture Hardware Assisted Rendering of Time-Varying Volume Data," *Proc. IEEE Visualization 2001 Conf.*, ACM Press, 2001, pp. 263–270.
- E. Lum, K.-L. Ma, and J. Clyne, "A Hardware-Assisted Scalable Solution of Interactive Volume Rendering of Time-Varying Data," *IEEE Trans. Visualization and Computer Graphics* vol. 8, no. 3, 2002, pp. 286–301.
- K.-L. Ma et al., "Parallel Volume Rendering Using Binary-Swap Compositing," *IEEE Computer Graphics & Applications*, vol. 14, no. 4, 1994, pp. 59–67.
- P. Lacroute, "Analysis of a Parallel Volume Rendering System Based on the Shear-Warp Factorization," *IEEE Trans. Visualization and Computer Graphics*, vol. 2, no. 3, 1996, pp. 218–231.
- K.-L. Ma and S. Parker, "Massively Parallel Software Rendering for Visualizing Large Scale Data Sets," *IEEE Computer Graphics & Applications*, vol. 21, no. 4, 2001, pp. 72–83.
- K.-L. Ma and D. Camp, "High Performance Visualization of Time-Varying Volume Data Over A Wide-Area Network," *Proc. Supercomputing 2000 Conf.*, CD-ROM, IEEE CS Press, 2000.
- M. Tory et al., "4D Space-Time Techniques: A Medical Imaging Case Study," *Proc. Visualization 2001 Conf.*, ACM Press, 2001, pp. 473–476.
- J. Kniss et al., "Volume Rendering Multivariate Data to Visualize Meteorological Simulation: A Case Study," *Proc. VisSym 2002*, 2002, pp. 189–194.
- A. Stompel, E. Lum, and K.-L. Ma, "Visualization of Multidimensional, Multivariate Volume Data Using Hardware-Accelerated Nonphotorealistic Rendering Techniques," *Proc. 10th Pacific Conf. Computer Graphics and Applications - Pacific Graphics 2002*, IEEE CS Press, 2002, pp. 294–402.
- J. Meredith and K.-L. Ma, "Multiresolution View-Dependent Splat Based Volume Rendering of Large Irregular Data," *Proc. 2001 Symp. Parallel and Large-Data Visualization and Graphics*, ACM Press, 2001, pp. 93–99.
- J. Rowlan, E. Lent, N. Gokhale, and S. Bradshaw, "A Distributed, Parallel, Interactive Volume Rendering Package," *Proc. Visualization '94 Conf.*, IEEE CS Press, 1994, pp. 21–30.
- K.-L. Ma, "Runtime Volume Visualization for Parallel CFD," *Proc. Parallel CFD '95 Conf.*, Elsevier, 1995, pp. 307–314.
- R. Haimes, "Unsteady Visualization of Grand Challenge Size CFD Problems: Traditional Postprocessing vs. Co-processing," *Proc. ICASE/LaRC Symp. Visualizing Time-Varying Data*, NASA Conf. Publication 3321, 1996, pp. 63–75.
- S.G. Parker and C.R. Johnson, "SCIRun: A Scientific Programming Environment for Computational Steering," *Proc. 1995 Supercomputing Conf.*, CD-ROM, IEEE CS Press, 1995.

**Kwan-Liu Ma** is an associate professor of computer science at the University of California, Davis. His career research goal is to improve the overall experience and performance of data visualization through more effective interactive techniques and user interface designs, expressive rendering, and high-performance computing. He has a PhD in computer science from the University of Utah. In 2000, he received the Presidential Early Career Award for Scientists and Engineers (PECASE) and Scholmbeger Foundation Award for his work in large data visualization. He has also served as a guest editor for several theme issues of *IEEE Computer Graphics & Applications*. He is a member of the IEEE, IEEE Computer Society, ACM, and ACM SIGGRAPH. Contact him at the Dept. of Computer Science, UC Davis, One Shields Ave., Davis, CA 95616; [ma@cs.ucdavis.edu](mailto:ma@cs.ucdavis.edu).