Executive Summary
Web-based High Performance Remote Visualization

Rhonda J. Vickery, Mississippi State University, rvickery@gri.msstate.edu
Andy Cedilnik, Kitware, Inc., andy.cedilnik@kitware.com
Joel Martin, James E. Fowler, Robert Moorhead, Mississippi State University
jmartin@gri.msstate.edu, fowler@ece.msstate.edu, rjm@gri.msstate.edu,
Yogi Dandass, Travis Atkison, Mississippi State University,
yogi@cse.msstate.edu, tla96@msstate.edu
Paul Adams, ERDC MSRC, paul.adams@erdc.usace.army.mil,
Jerry Clarke, US Army Research Laboratory, clarke@arl.army.mil

Introduction

This work investigates a web browser-based tool that allows interactive large data visualization and analysis from the desktop within the Department of Defense (DoD). Remote visualization (RMV) is requested more frequently than any other visualization service, and is often the first thing requested by a user with large dataset analysis requirements. RMV is technically possible, but a unified approach is needed across the DoD to make it successful, efficient, and useful. An interactive browser-based capability is the easiest to deploy to multiple desktop platforms, since no additional software beyond a browser is needed.

Objective

This system is designed to provide a simple web-based interface to visualization and analysis services for large data residing at the MSRC where it was produced.

Methodology

The WebVis component of ParaView Enterprise Edition is a framework to perform visualizations and data processing within a web browser-based environment. It contains project management tools as well as interactive previews of the running visualizations. The system allows users to run an arbitrary number of visualizations and visualization sessions, as well as generate images and animations. This system is also authenticated using the DoD Kerberos mechanism.

WebVis supports two types of users. The first is the data analysis expert or author, who creates visualization scenarios using desktop ParaView for WebVis viewers. The second type, the viewer, visualizes data using prepared scenarios through a basic browser-based interface. For example, the viewer may apply a pre-configured visualization to multiple different datasets or to monitor the results of running computational jobs on HPC systems.

Results

Recent enhancements have improved the performance and usability of the visualization component. In particular, the addition of two image compression schemes improves the image delivery to the user. This gives a choice of four total schemes for various combinations of remote computer, local computer, and bandwidth. We show that the new BISK compression scheme performs best for low bandwidth situations while SQUIRT works well for high bandwidth network connections. Encryption does not add a significant amount of overhead.

Significance to DoD

WebVis allows users to visualize and analyze large datasets without having to either move all the data to their local machine(s) or having to physically leave their office. Multiple users can view the same data at different remote locations. Users would not require high-end desktop analysis systems with large memory, disks, or powerful graphics cards but could still utilize HPCMP resources for the mapping from data to imagery.

Web-based High Performance Remote Visualization

Rhonda J. Vickery, Mississippi State University, rvickery@gri.msstate.edu
Andy Cedilnik, Kitware, Inc., andy.cedilnik@kitware.com
Joel Martin, James E. Fowler, Robert Moorhead, Mississippi State University
jmartin@gri.msstate.edu, fowler@ece.msstate.edu, rjm@gri.msstate.edu,
Yogi Dandass, Travis Atkison, Mississippi State University,
yogi@cse.msstate.edu, tla96@msstate.edu
Paul Adams, ERDC MSRC, paul.adams@erdc.usace.army.mil,
Jerry Clarke, US Army Research Laboratory, clarke@arl.army.mil

Abstract

This work describes a web browser-based remote visualization capability for large datasets. We discuss recent enhancements including access to databases and remote resources, as well as the addition of two image compression algorithms and their effect on performance. Results indicate that the existing SQUIRT image compression algorithm performs best for large bandwidth situations while the new BISK algorithm works better than the previous JPEG compression scheme. The results from a study on encryption effects on the data stream show that encryption does not add a significant amount of overhead.

1. Introduction

This work investigates a web browser-based capability that allows interactive large data visualization and analysis from the desktop within the Department of Defense (DoD). Remote visualization (RMV) is requested more frequently than any other visualization service, and is often the first thing requested by a user with large dataset analysis requirements. RMV is technically possible, but a unified approach is needed across the DoD to make it successful, efficient, and useful. An interactive browser-based capability is the easiest to deploy to multiple desktop platforms, since no additional software beyond a browser is needed.

There are a number of technologies available for remote visualization [1]. The key security feature that differentiates this application is that it follows specific High Performance Computing Modernization Program (HPCMP) security guidelines for Kerberos authentication plus additional encryption and security on the ticket granting ticket (TGT) [2, 3].

This work builds upon the results of an Army Research Lab funded SBIR phase II project with Kitware, Inc. that exploits ParaView. Additionally, this work has been coordinated with the User Interface Toolkit (UIT) effort [4]. The resulting browser-based product is part of ParaView Enterprise Edition (PVEE). The interactive performance of PVEE with additional encryption and authentication requirements has been investigated in order to facilitate the use of RMV by the DoD user community [5]. The emphasis has been on improving performance in the context of secure communications.

We present an overview of the WebVis application along with a description of several enhancements to improve the remote visualization experience. We describe the addition of two image compression algorithms and provide results on their performance impact. Finally we include some recommendations for future work and draw our conclusions.

2. Overview

The web visualization component of ParaView Enterprise Edition is a framework to perform visualizations and data processing within a web browser-based environment. It contains project management tools as well as interactive previews of the running visualizations. The system allows users to run an arbitrary number of visualizations and visualization sessions, as well as generate images and animations. Figures 1 and 2 (color plate) show common views from within WebVis.

The PVEE WebVis framework is based on several components (see Figure 3). These components include a web client, web server, database, User Connections Manager (UCM), Visualization Manager (VM), Server Runner (SR), and ParaView Backend Server (PBS).

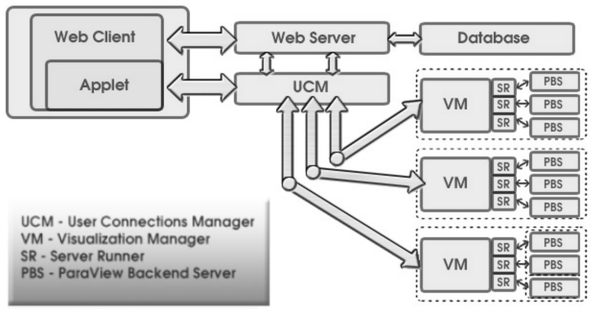

Web Client
Applet
Web Server
Database
UCM
VM
SR
PBS
UCM - User Connections Manager
VM - Visualization Manager
SR - Server Runner
PBS - ParaView Backend Server

Figure 3. WebVis Architecture

The Web Server performs all the necessary communication between the client and various server components, as well as the necessary authentication of the client. Once the client is authenticated, random keys are generated that are then used for the life of the user's session. The UCM secures incoming connections to the system and makes sure only connections that are authenticated are accepted. The UCM also routes the visualization traffic between the client and an appropriate VM. The VM is responsible for managing the running visualizations for a specific user (one per user). It is started when the user logs into the system and persists until the last visualization terminates. The UCM routes all of the user's visualization sessions to the VM, which sends the request to the actual visualization server through the use of a SR. The SR is a simple program that abstracts the running of visualizations for the site. The PBS performs all of the actual reading, processing, and visualizing of data. It is an extension of the regular desktop ParaView server that supports encryptions and image transfers appropriate for PVEE .

2.1. Usage Scenarios

It is important to distinguish between the two types of users that WebVis supports. The first is the data analysis expert or author, who creates visualization scenarios for WebVis viewers. The viewer may or may not be a data analysis expert, but needs to visualize data using prepared visualization scenarios through a basic browser-based interface. For example, the viewer may apply a pre-configured visualization to multiple different datasets or to monitor the results of running computational jobs on HPC systems.

The author develops visualizations using desktop ParaView then exports it to an XML-based format. He then logs into the WebVis service and uploads the exported visualization to the web server. He then configures the interface to make visualizations available to specific viewer groups and choosing which GUI options are needed. Viewers may then access the WebVis service to run visualizations available to them and modify parameters based on GUI options made available by the author. In this way the interface can be streamlined to the type of visualization required for analysis.

3. Recent Improvements

We describe several improvements to PVEE which include an infrastructure for access to databases, access to remote resources, and the addition of two image compression algorithms.

3.1. Web Client Server Interactions

To reduce the latency overhead we made significant changes to the web client to server interactions in the areas of mouse interaction and updating web pages. We improved the responsiveness of the render window while interacting by using a pipelining approach. Since compression schemes require additional computation on the client side to decode and display images, we cache the most recent mouse interaction event while waiting for the latest response from the server. Once it arrives we send the cached event to the server so that the next image can be rendered while the client is decoding the image just received. Mouse interaction events are immediately sent to the server if no server interaction responses are pending.

We transformed the web client to use the "AJAX" (Asynchronous JavaScript and XML) web technology. The concept in AJAX systems is to only send the changes to the client. Thus, the entire web page is rarely, if ever loaded. Only the sections that change due to the operations that the user requested are updated. This not only reduces the server side effort in generating the client response but also improves the user experience, since the entire page is no longer updated. To support an AJAX web client, the web server also needs to be modular, generating responses only for the changed sections. As a result, the server workload is reduced, since it does not regenerate redundant sections. Also, we implemented deferred validation of the running states of the visualization servers. Previously every web server request would generate an action to verify that each visualization server was indeed running. Now we check to see if the visualization server is running only when the user requests access to it.

3.2. Database Access

We have developed a unified framework to access data in a database. As part of this effort, all information about the user session is now stored in the database. The only persistent objects that are not in the database are the actual datasets, visualization states, images, and animations. These object database entries contain location information.

ParaView uses a MySQL database for its internal data storage requirements. One major advantage of using a database system over a traditional file based approach is efficient data access to query information. The second is to allow concurrent access by multiple web clients through the database locking mechanism. Internally, WebVis uses the object model adaptor to represent every entry in the database by an individual object. This object concept makes retrieving, manipulating, and storing data efficient. WebVis can also access external databases through the XML-RPC interface system or through the internal object-model interface.

An example object in the WebVis code is the currently running visualization. This object contains information about the state of the current visualization, and values retrieved from other tables, such as the original visualization that started it, the user information for the user who created the visualization, and the user that runs the visualization. WebVis isolates the database dependent code to a single module. This makes it convenient to replace the database with other implementations such as SQLite, or even traditional files.

Switching to the database significantly improved WebVis performance. The improvements were the result of using optimized database access as opposed to the flat file access. In addition, the new database code uses internal database joins to extract related information, while the previous code first loaded all of the data into memory and then joined it. Finally, to retrieve data from the flat files, most of the file had to be read to retrieve small pieces of information. In the new design only the necessary rows need to be read.

3.3. PVEE Access to Remote Resources

In the most general case, the VM can reside anywhere that the UCM can access it. This can be on the web server, on the head node of the rendering cluster, or on a dedicated system. The actual location may depend on several factors, such as network topology, size of the system, and the network traffic load. For example, several cluster installations only provide network access to the first node. In this case the VM has to run on the first node of the cluster. Alternatively, if any node on the cluster can be accessed, then the VM can be located anywhere with network access to the UCM and the cluster.

When accessing remote locations, several preconditions have to be met. First, the remote location has to be accessible over the network. Second, the user has to be able to login to the remote system. Finally, the user has to be able to remotely launch processes (such as via SSH or RSH). An additional limitation in the current implementation is that only a single VM can be used between the UCM and PBS. This restricts the network topologies on which the PVEE WebVis component can be implemented.

Consider an example scenario with two sites. The first site hosts all of the managing resources and some clusters. The second site hosts several clusters. The web server as well as the UCM can be set up at the first site. In the VM configuration, each cluster has access to some VM. This can be one VM per cluster or one VM per site.

3.4. Image Compression Algorithms

Previously there were two modes of image compression implemented within WebVis: SQUIRT and JPEG. We describe these along with two new algorithms: BISK and DPCM. We show why BISK is best for low bandwidth situations, while SQUIRT remains the best choice for high bandwidth scenarios.

3.4.1. SQUIRT Compression

SeQuential Unified-channel Image Run Transmission (SQUIRT) is a simple image compression approach that was developed by Sandia National Laboratories. It can obtain rates of 7-8 Hz at 1280x1024 resolutions over 100-1000 Mb/s links.

SQUIRT is a compression technique that works well for lossy and lossless compression and is targeted for fast links of at least 100 Mb/s. Existing compression techniques take far too long to compress mega-pixel images and for high speed links too much time is wasted compressing the images. It is faster to just send the images uncompressed over the network link.

SQUIRT takes about 1/100 of a second to compress or decompress images by keeping the RGBA channels unified into 4-byte word boundaries and then using bitmasks to enable a lossy run-length encoding of the image. The bitmasks have the additional advantage of allowing different loss rates on the color channels so the current bitmasks have the least 'loss' on the green channel, since the eye is most sensitive to changes in green.

The current implementation of PVEE uses the SQUIRT algorithm as one of the available compression schemes. In order to keep the channels 'word aligned,' the color buffer must be RGBA. The compression routine then packs the run length into the alpha channel, discarding any alpha information. The decompression component then simply pulls the run from the alpha channel and sets alpha to one. So the receiver can do a "write pixels" with the RGBA format. Sandia developed SQUIRT to be utilized for ParaView client-server image delivery. It also performs sufficiently for WebVis interaction.

3.4.2. JPEG Compression

JPEG compression is based on the JPEG image file format. Each frame is compressed in memory and then sent over the network. On the client side, the image is decompressed and displayed. The speed performance of JPEG compression is not as good as the performance of SQUIRT compression. However, the size of JPEG compressed images is much smaller, which is useful for slow connections. For example, an image of size 450x450 pixels is about 800,000 bytes uncompressed. The SQUIRT compressed image is only 40,000 bytes, which is a significant improvement from the original image size. JPEG compression produces an even smaller image size of 20,000 bytes.

3.4.3. BISK and DPCM Compression

The two compression algorithms recently added to PVEE are differential pulse code modulation (DPCM) [6] and binary set splitting with k-d trees (BISK) [7]. On the server side, both the DPCM and BISK encoders first convert the RGB image to YCbCr color space, and then the two chrominance components are down sampled to one-quarter of their original size. The DPCM encoder encodes each image component independently by first predicting the current pixel, scalar quantizing the difference between the current pixel and its prediction, and then applying an exponential Golomb entropy coder [8] to the sequence of quantizer-indices. The BISK encoder encodes each image component independently by first applying an integer-valued 5/3 wavelet transform [9] and then applying the BISK algorithm [7] to encode the wavelet coefficients.

Figure 4a (color plate) shows the original RGB image passed to the encoders, and Figures 4b–4e show the corresponding reconstructed image at the client-side decoder for the two new compression algorithms, DPCM and BISK, as well as for the two algorithms already existing in PVEE, SQUIRT and JPEG. The images in Figures 4b–4e were produced with encoder parameters set such that all encoders produced approximately the same image quality as measured by a signal-to-noise ratio (SNR) of 16 dB (this is a color-image SNR based on the 1964 CIE formula for distance between two colors as calculated in the CIE Modified UCS color space) [6]. SQUIRT, being a very computationally lightweight algorithm, achieves the faster encoding and decoding times. However, the rate-distortion performance of the other three algorithms is substantially superior to that achieved by SQUIRT - the DPCM, JPEG, and BISK algorithms require a much smaller rate (fewer bits per pixel (bpp)) to achieve the same image quality as shown in Figure 5 with an expanded view of the BISK and JPEG rate performances in Figure 6.

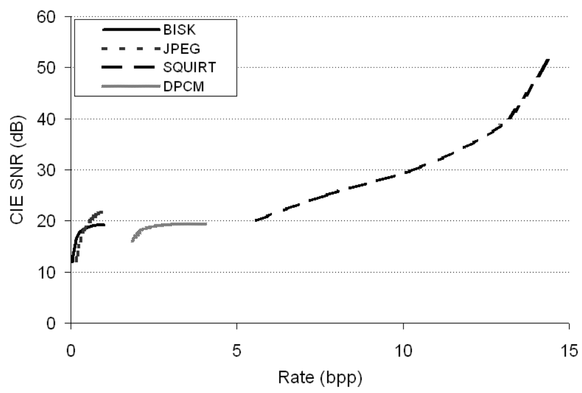


Figure 5. Rate-distortion performance of all four coders.

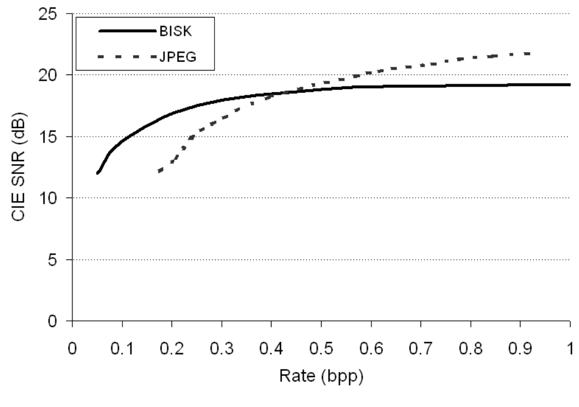


Figure 6. Rate-distortion performance of BISK and JPEG coders.

BISK (Figure 4a) achieves a rate somewhat smaller than JPEG, but is slightly more computationally complex. Both BISK and JPEG perform better than DPCM, although the JPEG images tend to show blocking artifacts at low rates. Overall, SQUIRT might be the preferred algorithm in situations in which the network bandwidth to the client is large, while BISK would be preferred when tight communication-bandwidth constraints are imposed. The DPCM algorithm, on the other hand, does not appear to offer any advantages, since both JPEG and BISK provide better compression efficiency and are faster.

3.5. Encryption Effects on Compression Study

For this study we investigated the effect of encrypting visualization-related packet streams on interactivity. The time T taken to transmit an image over the network is T = L + D/B, where L is the latency, D is the size of the data to be transmitted, and B is the bandwidth of the communication infrastructure. The bandwidth of the system is fixed and is assumed to be large, making the contribution of the second term (i.e., D/B) smaller than the contribution of the first term (i.e., latency). Therefore, it is critical to reduce the latency in the WebVis data stream.

There are several contributing factors to the overall latency in image transmission. Of these factors, the latency inherent in data transmission (e.g., per-hop store-and-forward frame and packet formatting and processing, error detection, and signal propagation delay) is fixed. However, the latency caused by WebVis itself varies depending on how the various components are implemented and distributed across multiple computer systems. Adding Kerberos-based security and encryption will contribute additional latency in the multi-layered implementation of WebVis. Latency can be measured quantitatively using quality-of-service (QoS) metrics in terms of frame-rate (frames per second) and qualitatively in terms of responsiveness to user commands for rendering different views. Therefore we measured the effect of encrypting the WebVis data stream on interactivity in terms of frame-rate and request latency.

To test the theoretical performance of WebVis, four tests were conducted on the local network at

Kitware with one user on a polygonal dataset with approximately 200 polygons. Note that the latency of WebVis being measured is heavily dependent on the image size regardless of the visualization hardware available for the PBS. In this case the entire WebVis architecture was running on a basic desktop Linux PC, and the relative differences due to compression schemes and encryption were studied.

Each test used the same server and client and they all tested the effect of changing the sending image size to the performance. Each of the four tests were run 50 times and averaged. Since all systems were on the same network, this simulates near peak performance of the system. However, there is still substantial room for improvement.

The four tests were: JPEG compression with encrypted data stream (T1), JPEG compression with unencrypted data stream (T2), SQUIRT compression with encrypted data stream (T3), and SQUIRT compression with unencrypted data stream (T4). The measurements taken were frames per second (fps), compressed image size, time for the server to respond to the request, and time for the client to decode the image (See Figures 7-9). The original image resolution was 450x450 pixels, and the image size divisor is the parameter used to decrease image size (the X axis variable in the figures). For example, an image size divisor of 1.5 corresponds to an image size of 450/1.5, or a resolution of 300x300.

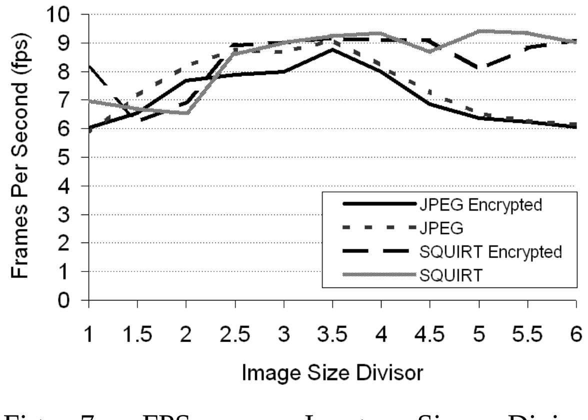


Figure7. FPS vs. Image Size Divisor Measurement

As visible in Figures 7 and 8, both encrypted and unencrypted streams perform similarly. The conclusion is that encryption adds an almost insignificant amount of processing. Figure 9 also indicates that client decoding time is similar for both encrypted and unencrypted streams. However, different compression schemes provide different results. JPEG compression produces smaller images, so it is much better for slow connections. SQUIRT, on the other hand, requires much less processing on the client and server to perform the compression, so it is much more effective for fast connections.

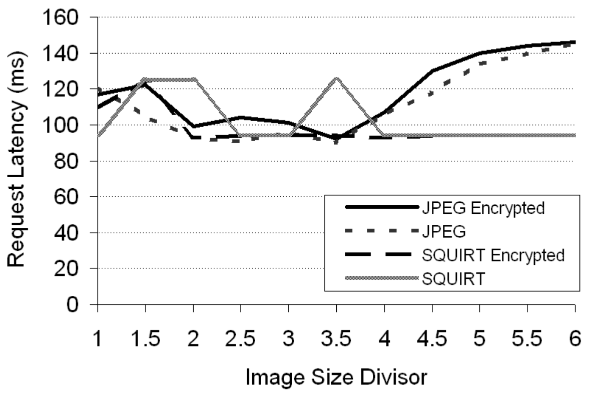


Figure 8. Request Latency vs. Image Size Divisor Measurement.

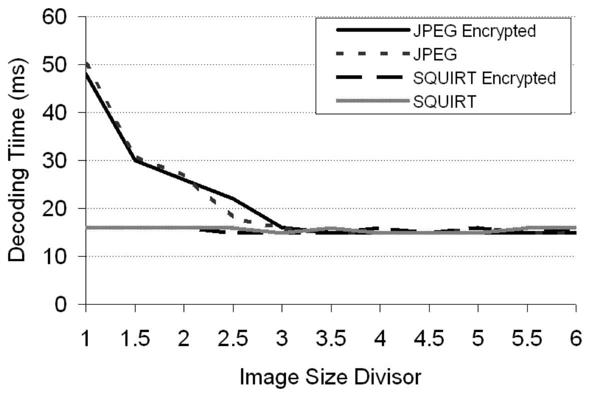


Figure 9. Decoding Time vs. Image Size Divisor Measurement.

Subdivision also adds improvement in performance with degradation of the image quality. Therefore, after subdividing the image more than three times, the performance improvement becomes less significant and the image size does not decrease enough to justify the lower image quality.

Using a Linux PC as a WebVis PBS server would not be the ideal platform for deployment. Given the low-end hardware used in this study, the user performance is still acceptable.

4. Conclusions

We described several improvements to WebVis that reduce latency and generally improve the overall experience. We also implemented two new algorithms for image compression. Of these, BISK appears to be slightly more computationally complex than JPEG for lower bandwidth situations, but provides better image quality. SQUIRT remains the best compression algorithm for higher bandwidth network connections. The effect of encryption on the WebVis image streams was studied and we concluded that encryption does not add a significant amount of overhead. In addition, we are in the process of gathering additional performance numbers from multiple locations under several configurations that users may encounter. From these we expect to show how effective WebVis will be for a DoD user for everyday analysis. We look forward to continuing to help users find useful ways of analyzing their data. WebVis provides a way to streamline the visualization process and access the results through a simplified browser-based interface.

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United Stated Air Force, Department of Defense, United States Government, or Mississippi State University. Portions of this work were made possible through support provided by DoD HPCMP PET activities through Mississippi State University under contract No. GS04T01BFC0060.

References

[1] R. J. Vickery, "Remote Visualization: What It Could Be in the DoD HPCMP" in *Aeronautical Systems Center Major Shared Resource Center Wright Cycles* 2005, pp. 22-23.

[2] "High Performance Computing Modernization Program Guidelines for Implementing Secure Access to HPCMP Systems," http://www.hpcmo.hpc.mil/Htdocs/SECURITY/dren_secure_access_guidelines.pdf, 2004.

[3] J. Garman, *Kerberos: The Definitive Guide*. Sebastopol, CA: O'Reilly & Associates, Inc., 2003.

[4] P. Duett, W. Monceaux, and K. Rappold, "EZHPC: Easy Access to High Performance Computing," in *Proceedings of the HPCMP Users Group Conference*, 2004, pp. 289-292.

[5] R. J. Vickery, A. Cedilnik, J. P. Martin, Y. Dandass, T. Atkison, R. J. Moorhead, J. Clarke, and P. Adams, "Web-based Secure High Performance Remote Visualization," in *Proceedings of the SciDAC Conference*, 2006.

[6] A. K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prentice Hall, 1989.

[7] J. E. Fowler, "Shape-Adaptive Coding Using Binary Set Splitting with k-d Trees," in *Proceedings of the International Conference on Image Processing*, Singapore, 2004, pp. 1301–1304.

[8] J. Teuhola, "A Compression Method for Clustered Bit-Vectors," *Information Processing Letters*, vol. 7, pp. 308–311, October 1978.

[9] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Lossless Image Compression Using Integer to Integer Wavelet Transforms," in *Proceedings of the International Conference on Image Processing*, Lausanne, Switzerland, 1997, pp. 596-599.

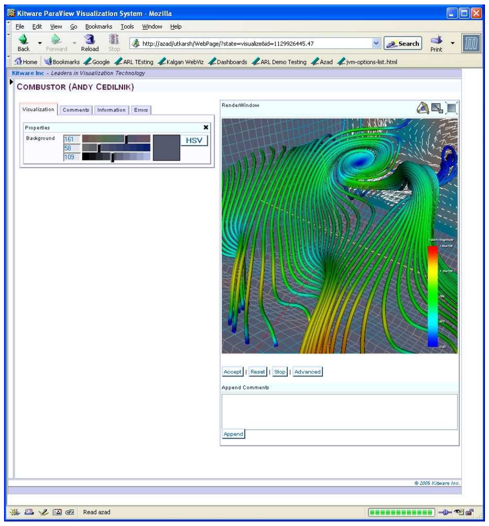


Figure 1. Example Visualization With Editable Fields and Annotations. Unrestricted image generated from publicly available ParaView example data.

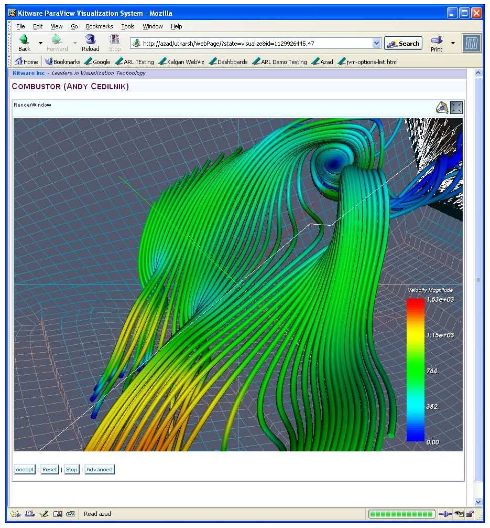
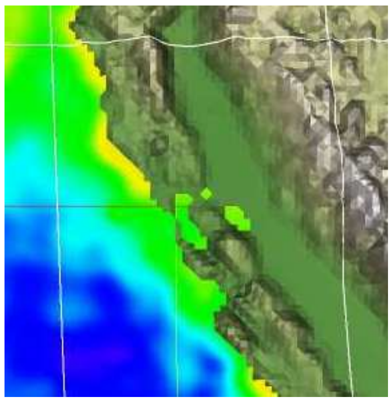
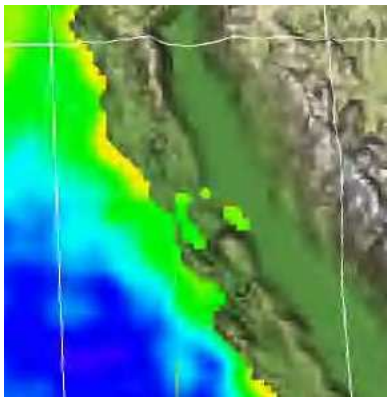
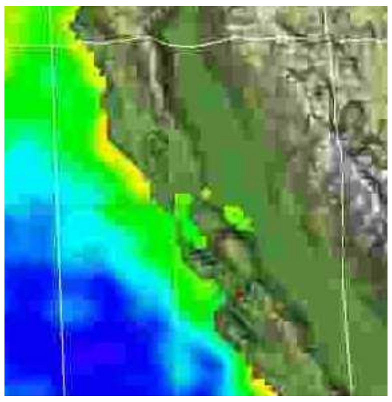
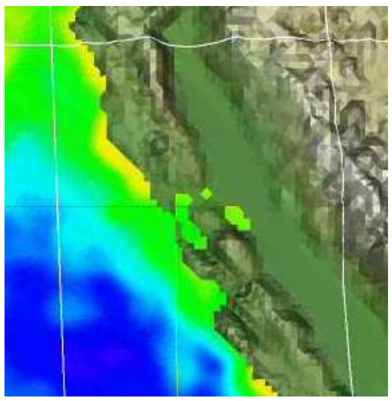


Figure 2. Full Screen Visualization Render Window. Unrestricted image generated from publicly available ParaView example data.


(a) Original Image.


(b) BISK Coding.


(c) JPEG Coding


(d) DPCM Coding.

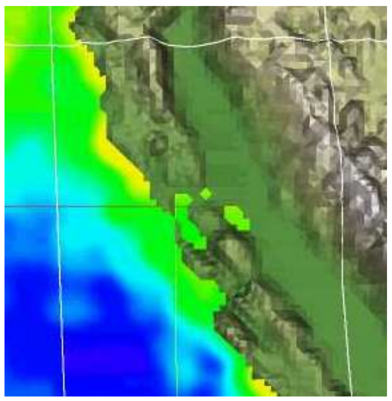

(e) SQUIRT Coding.

Figure 4. Comparison of original image (a) with images compressed with BISK (b), JPEG (c), DPCM (d), and SQUIRT (e) coders. Unrestricted visualization images courtesy of Derek Irby, Mississippi State University, of Coupled Ocean Atmosphere Mesoscale Prediction System data from Rick Hodur (NRL-MRY). Image analysis performed by James E. Fowler.