Feature Based Volumetric Video Compression for Interactive Playback

Bong-Soo Sohn[*] Chandrajit Bajaj[†] Vinay Siddavanahalli[‡]

Department of Computer Sciences & TICAM
University of Texas at Austin

Abstract

In this paper, we describe a compression scheme for encoding time-varying isosurfaces and amorphous volumetric features (volumes within specified value ranges) in a unified way, which allows for on-line reconstruction and rendering. Since the size of even one frame in a time-varying data set is very large, transmission and on-line reconstruction are the main bottlenecks for interactive visualization of time-varying volume and surface data. To increase the run-time decompression speed and compression ratio, we decompose the volume into small blocks and encode only the significant blocks that contribute to the isosurface and volumetric features. The result shows that our compression scheme achieves high compression ratio with fast reconstruction, which is effective for client-side rendering of time-varying isosurfaces with amorphous volumetric features.

Keywords: Compression, Wavelet Transform, Time-Varying Volume Visualization, Hardware-Acceleration, Isocontouring.

1 Introduction

As computing power and scanning precision rapidly increases, scientific simulations and measurements generate more and more densely sampled time-varying volume data which have very large sizes. For example, the size of an oceanographic temperature change data set tested in this paper is 237MB/frame ($2160 \times 960 \times 30$ float type) $\times$ 115 frames, and the gas dynamics data set is 64MB/frame ($256 \times 256 \times 256$ float type) $\times$ 144 frames. To visualize these data, *volume rendering* and *isocontouring* are performed frame by frame, so that a user can navigate and explore the data set in space and time. Each rendering technique has its own strength. For example, while volume rendering can display amorphous volumetric regions specified by the transfer function with transparency, isocontouring can provide the geometric shape of the surface specified by an isovalue. Both techniques are often combined for better understanding of the overall structure in a volumetric data set.

While current state-of-the-art graphics hardware allows very fast volume and surface rendering, transfer of such large data between data servers and browsing clients can become a bottleneck due to the limited bandwidth of networks. Therefore, an efficient data management scheme is an important factor in the rendering performance. To reduce the size of the data set, it is natural to exploit temporal and spatial coherence in any compression scheme. However, since the data size of even a single frame is very large, run-time decompression can be a bottleneck for interactive playback.

From this motivation, we developed a unified compression scheme for encoding time-varying isosurfaces and associated volumetric features. The inputs to compress a temporal sequence of isosurfaces and associated volumetric features are k isovalues $iso_i, i = 1..k$, l value ranges $[r_a, r_b]_i, i = 1..l$ and discretized time-varying volume data V. The data V, containing T time steps can be represented as $V = \{V_1, V_2, ..., V_T\}$, where $V_t = \{f^t_{i,j,k} | i, j, k$ are indices of x, y, z coordinates$\}$ is the volume at time step t, containing data values $f^t_{i,j,k}$ at the indices i, j, k. Our primary goal is to

- compress time-varying isosurface and volume features in a unified way.
- reduce the size of time-varying volumetric data with minimal image degradation.
- allow real-time reconstruction and rendering with PC graphics hardware acceleration.

We are going to borrow the idea of MPEG compression to efficiently exploit spatial and temporal coherence in data sets. However, direct extension of MPEG for 2D video compression to time-varying isosurface and feature compression is not suitable for satisfying our goals. Since MPEG encodes and decodes every block in each frame of the volumetric time-varying data, unnecessary regions within the data may also be encoded and decoded.

We adopt a block-based wavelet transform with temporal encoding in our compression scheme. The wavelet transform is widely used for 2D and 3D image compression. By truncating insignificant coefficients after wavelet transformation, these schemes achieve high compression ratio while keeping minimal image distortion. However, complete transformation of each frame is a waste of space and time resources, because function values not contributing to the given isosurfaces and volumetric features do not need to be encoded. In addition, the contributing values which have small changes over time do not need to be updated. Therefore encoding and decoding only the values significant in space and time instead of the full volume can improve both compression ratio and decompression speed.

Each frame of volume is classified as either an intra-coded frame or a predictive frame. The intra-coded frames can be decompressed independently while the predictive frames are the differences from the previous frame. Assuming that different blocks have different temporal variance, we can sort the blocks based on their temporal variance and truncate insignificant blocks to achieve higher compression ratios and faster decompression speeds.

In addition to efficient compression, fast reconstruction and rendering of the isosurface and volumetric features are also achieved. By attaching seed cells in the compressed stream of each volume frame, the rendering browser can construct the isosurfaces in minimal time. The fast speed comes from removing the search phase for finding at least one cell intersecting with each isosurface component. Since the isosurface is fixed, we need to store only one seed cell per isosurface component, which hardly affects the compression ratio. We can also compress the selected set of components

[*]bongbong@cs.utexas.edu
[†]bajaj@cs.utexas.edu
[‡]skvinay@cs.utexas.edu

in isosurfaces and volumes, and their evolution by using the feature tracking method [Silver and Wang 1997]. The reconstructed features can be rendered in real-time using PC graphics hardware.

Figure 1 shows the overall architecture of our real-time volumetric video display system. The algorithm requires the features to be defined before compression. These features can be identified manually or by automatic feature detection tools [Pfister et al. 2001]. To save disk storage space and to overcome the limitation of I/O bandwidth in network systems, a series of compressed frames are read from data source servers to browsing clients. Once each compressed frame is read, it is decompressed in software. The reconstructed image array is used for accelerated isocontouring and also sent to the texture memory in the graphics hardware for displaying volumetric features. This architecture can allow users to explore and interact with isosurfaces embedded in the amorphous volumetric features in space and time, which is our ultimate goal.

The remainder of this paper is organized as follows. First, the related work is described. Then, in section 3, volumetric video displaying architecture is described. In section 4, volumetric video compression scheme supporting interactive decompression is proposed. Section 5 describes our scheme for interactive browsing of compressed time-varying features. Experimental results are described in section 6. Finally, in section 7, we give a conclusion.

2 Related Work

Visualization of time-varying volume data has been a challenging problem. Compression, time-based data structures, and high performance visualization systems have been introduced to cope with overwhelming data sizes and heavy computation requirements.

Compression Compression is extremely useful for large data manipulation, especially for transmission of data from servers to browsing clients. Since scientific data tend to be very large and have lots of redundancy, people prefer to use compressed data for efficient use of the memory and I/O bandwidth. A number of algorithms for image and surface compression have been developed.

Papers on single resolution and progressive compression of triangulated surfaces (e.g. isosurfaces) include those by [Taubin and Rossignac 1998; Khodakovsky et al. 2000]. A compression scheme specialized for isosurface [Zhang et al. 2001] utilizes the unique property of an isosurface that only the significant edges and function values defined on a vertex are required to be encoded.

Most of image compression techniques are geared towards achieving the best compression ratio with minimal distortion in the reconstructed images. JPEG and MPEG [Gall 1991] are developed for compressing still images and 2D video data with controllable size and distortion tradeoff. Embedded coding algorithms such as embedded zero tree wavelet(EZW) [Shapiro 1993] and set partitioning in hierarchical trees(SPIHT) [Said and Pearlman 1996] are useful for progressive transmission and multimedia applications.

For 3D image compression, Ihm et al. [Ihm and Park 1998] described a wavelet-based 3D compression scheme for visible human data and later extended it to 3D RGB image compression for interactive applications such as light-field rendering and 3D texture mapping [Bajaj et al. 2001b]. Compression ratio can be improved by capturing and encoding only significant structures and features in the data set [Bajaj et al. 2001a; Machiraju et al. 1998]. In those compression schemes the primary goal is fast random access to data, while maintaining high compression ratios.

[Guthe and Straser 2001] applied the MPEG algorithm to time-varying volume data using wavelet transformations. They compare the effects of motion compensation and the usage of different wavelet basis functions. [Lum et al. 2001] exploits texture hardware for both rendering and decompression. Since data is transferred in the compressed format between different memory systems, I/O time is significantly saved.

Time Based Data Structure Time-Space Partitioning(TSP) Tree is introduced and accelerated later by using 3D texture mapping hardware [Ellsworth et al. 2000] for fast volume rendering of time-varying fields. The efficiency comes from skipping insignificant rendering operations and reusing the rendered images of the previous time step.

[Shen 1998] proposed the Temporal Hierarchical Index (THI) Tree data structure for single resolution isocontouring of time-varying data, by an extension of his ISSUE algorithm [Shen et al. 1996]. The THI tree provides a compact search structure, while retaining optimal search time. Hence, expensive disk operations for retrieving search structures are reduced. Sutton et.al. [Sutton and Hansen 1999] proposed Temporal Branch-on-Need Tree by extending octrees for minimizing unnecessary I/O access and supporting out-of-core isosurface extraction in time-varying fields.

[Shamir et al. 2000] developed an adaptive multi-resolution data structure for time dependent polygonal meshes called *T-DAG*(Time-Direct Acyclic Graph). *T-DAG* is a compact representation which supports queries of the form *(time-step,error-tol)*, and returns an approximated mesh for that time step, satisfying the error tolerance.

High Performance Visualization System [Ma and Camp 2000] describes a remote visualization system under the wide area network environment for the visualization of time-varying data sets. Current state-of-the-art graphics hardware enables real-time volume and isosurface rendering [Westermann and Thomas 1998] and decompression [Lum et al. 2001].

Isosurface Extraction A large amount of research has been devoted in the past for fast isosurface extraction from 3D static volume data. The Marching Cubes algorithm [Lorensen and Cline 1987] visits each cell in a volume and performs appropriate local triangulation for generating the isosurface. To avoid visiting unnecessary cells, accelerated algorithms [Cignoni et al. 1997] minimizing the time to search for contributing cells are developed.

The contour propagation algorithm is used for efficient isosurface extraction [Bajaj et al. 1996; Howie and Black 1994]. Given an initial cell that contains an isosurface component, the remainder of the component can be traced by contour propagation. This property significantly reduces the space and time required for searching cells containing the isosurface by using a small number of seed cells. Multiresolution [Gerstner and Pajarola 2000] and view-dependent techniques [Zhang et al. 2002] are useful to reduce the number of triangles in an isosurface.

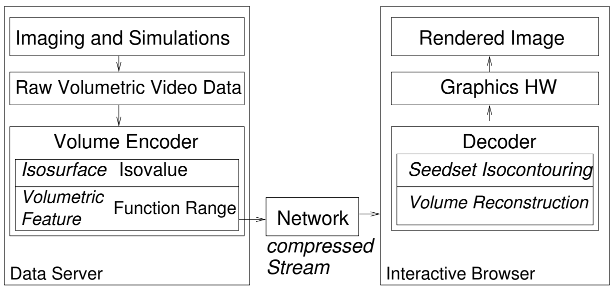

Figure 1: Volumetric video displaying process

3 Volumetric Video System

Like widely used 2D video systems, a *volumetric video* system displays a sequence of 3D images over time, frame by frame. While in a 2D video, users can only look at continually updated 2D images in a passive way, a volumetric video, or time-varying volume visualization system allows them to explore and navigate the 3D data in both space and time. Considering that most scientific simulations generate dynamic volume data, volumetric video systems are especially helpful for scientific data analysis.

A naive way for displaying time-varying 3D volume data is to repeat reading each frame from the data server and rendering the volume with the given visualization parameters. Since most time-varying scientific data sets are very large and have high spatial and temporal coherence, it is natural to apply compression for reducing storage overheads and transmission times. However, run-time decompression of data encoded by standard static and time-varying image compression schemes may become a bottleneck in real-time playback of volumetric video because they usually decompose an image into blocks and decode every block during decompression. During compression, we order the blocks based on their significance and encode only significantly changing blocks. This increases the run-time decompression speed as we have limited the number of blocks to decode.

A two-stage strategy is adopted to enable interactive navigation and exploration of very large time dependent volume data. In the first stage on the server side, the large time-dependent volume is analyzed and processed on a high performance server so that results of this volumetric processing is an intermediate multi-resolution, time-dependent volumetric representation of interesting features (isosurface, volume within a range) of the data, generated and stored in a compressed format. The intermediate multi-resolution representation permits tradeoffs between interactivity and visual fidelity for the second, interactive browsing stage. In the second stage on the client side, the volumetric video is decoded and played back by an interactive visualization browser that can be made available on a standard desktop workstation equipped with a 3D graphics card. In contrast to a standard video player, the visualization browser can allow certain levels of interactivity such as dynamically changing viewing parameters, modifying lighting conditions, and adjusting color-opacity transfer functions, in addition to timed playback of the volumetric video along with some user-specified fly-path in space and time.

4 Compression Scheme

In this section, we describe a unified scheme for compressing both time-varying isosurfaces and volumetric features at the same time. By encoding only significant function values based on associated weights using a wavelet transform, we can achieve high compression ratios. However, simple function encoding requires on-line isosurface extraction in the client side. To accelerate this surface extraction process, we insert seed cells into the compressed volume frames. In the following sections we use the term *feature* to mean either an isosurface, or a volumetric feature specified by a scalar value range.

4.1 Compression

The input data set is a time dependent volume data set, $V = \{V_1, V_2, ..., V_T\}$ with k isovalues $iso_1, ..., iso_k$ and l value ranges $r_1 = [r_a, r_b]_1, ..., r_l = [r_a, r_b]_l$. Each frame of the volume is classified as either an intra-coded frame or a predictive frame. For each isovalue and range, the reconstructed quality can be specified as a threshold value for wavelet coefficients and another threshold value for the blocks in predictive frames, such that

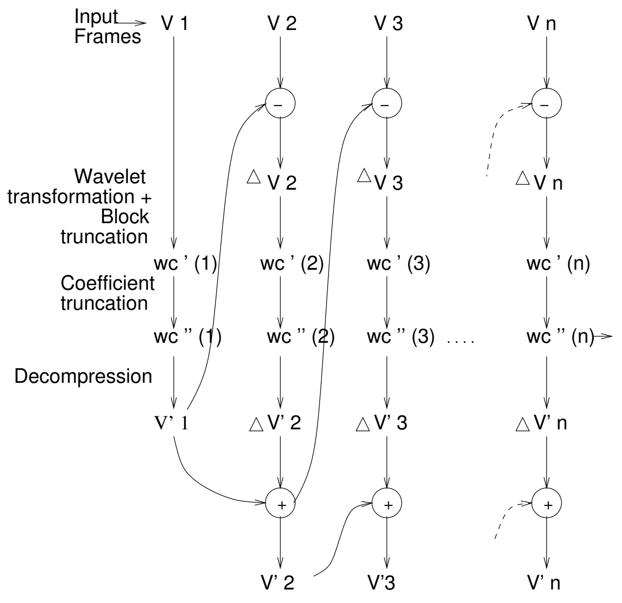

Figure 2: The overall compression algorithm

wavelet coefficients or blocks not satisfying that value are truncated. The criteria for truncation is given in the steps of the compression algorithm below. The whole data set is represented as $V = \{\{I_1, P_{11}, P_{12}, ..., P_{1p_1}\}, ..., \{I_N, P_{N1}, P_{N2}, ..., P_{Np_N}\}\}$ where I_i is an intra-coded frame in the $i-th$ temporal group and P_{ij} is the $j-th$ predictive frame in the $i-th$ temporal group.

Assuming that there are only small changes between consecutive frames, wavelet transformation of changes instead of the entire frames yields higher compression ratios and lower decompression times. Therefore, the compression of an intraframe is independent of other frames while compression of a predictive frame is dependent on previous frames in the same temporal group. The overall compression algorithm is shown in figure 2. Note that compression is performed on each volume and only significant values contributing to the features are encoded. All 3D frames are decomposed into $4 \times 4 \times 4$ blocks and wavelet transformation is performed on each block contributing to the specified features in the volume. The steps of the compression algorithm are as follows:

1. **difference volume**: $\Delta V_k = V_k - V'_{k-1}$, where V_k is original image of k-th frame and V'_{k-1} is the reconstructed image of the compressed volume V_{k-1}. If V_k is an intra-coded frame, we assume V'_{k-1}=0.

2. **wavelet transformation**: $W\Delta V_k$ = wavelet transformation of ΔV_k. Compute coefficients $c_1, ..., c_m$ representing ΔV_k in a three-dimensional Haar wavelet basis.

3. **classification** Each wavelet coefficient c and block b is classified as either insignificant or significant. c and b are further classified based on which features they contribute to. c and/or b can belong to more than one feature. In such cases, the survival of c or b is dependent on the highest weighted feature that contains them.

4. **truncation of blocks** A block which does not contribute to the features or has very small changes over time is considered as an insignificant block. By truncating insignificant blocks, we can achieve higher compression ratios and can control the time for the volume reconstruction. To identify blocks contributing to the i-th feature and having insignificant

changes, the sum of the square of coefficients is compared with a threshold value λ_i. If the sum is less than λ_i, the block is truncated. For encoding the truncated block, only one bit is assigned in the block significance map.

5. **truncation of wavelet coefficients** : The i-th feature to be compressed has its own weight represented as a threshold value τ_i. By setting the threshold value, the reconstructed quality of a specific feature can be controlled. If a wavelet coefficient c associated with the i-th feature is less than τ_i, the coefficient is truncated into zero.

6. **encoding**: The overall encoding scheme is shown in figure 3. Once wavelet coefficient truncation is performed based on each features weight, we take the surviving coefficients and encode them. The encoding is performed on each block, resulting in a sequence of encoded blocks. We classify 64 coefficients in a block as one level-0 coefficient, 7 level-1 coefficients and 8×7 level-2 coefficients to take advantage of the hierarchical structure of a block.

   In the header of a frame, a bit stream representing each block's significance is stored to indicate whether the block corresponding to each bit is a zero-block or not.

   This avoids additional storage overhead for insignificant blocks. One bit is assigned to each block in sequence. Then, for each significant blocks in sequence, we store an 8bit map representing whether the one level-0 and seven level-1 coefficients are zero or not. Next, 8bit map representing whether each eight $2 \times 2 \times 2$ subblock has non-zero wavelet coefficients followed by significance map for representing non-zero level-2 coefficients. After storing level-2 coefficient significance maps, actual values of non-zero wavelet coefficients are stored in order. We used two bytes for storing a coefficient value.

   Lossless compression is further applied to improve compression.

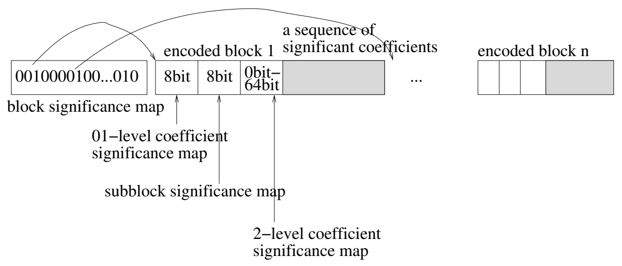

Figure 3: Suggested encoding scheme for supporting fast decompression and high compression ratios

4.2 Seed Cells Insertion

To allow browsing clients to quickly render isosurfaces encoded in a volume, seed cells are attached to the compressed stream of each frame. A seed cell is guaranteed to intersect with a connected component of the isosurface. By performing surface propagation from the given seed cells, we can avoid visiting unnecessary cells and save extraction time. Since only one seed cell per isosurface component is necessary, the size of the seed cells set is negligible and search structures such as octrees and interval trees are not required. Therefore, the isosurface extraction time is only dependent on the number of triangles regardless of the volume size.

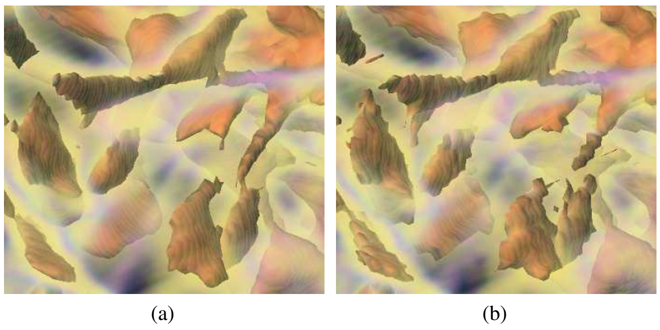

Figure 4: Performance comparison of different encoding schemes: Although the compression ratio of (a) and (b) is same (181:1), the quality of the reconstructed image using (a) is much better than (b). (a) Only the values contributing to the isosurface and volumetric features are encoded. (b) Every value in the volume is encoded.

4.3 Run-Time Decompression

Since we have a sequence of wavelet encoded volumes $W\Delta V_k$, we can get an approximated image V_k' by decoding and performing an inverse wavelet transformation. More specifically, $\Delta V_k' = inverse\ transformation(W\Delta V_k)$ and $V_k' = V_{k-1}' + \Delta V_k$. For the intra-coded frame V_i, $V_{i-1}' = 0$ and we can get V_i' directly from $\Delta V_i'$ with no dependency on other frames. Once V_i is reconstructed, succeeding predictive frames can be decoded frame by frame until the next intra-coded frame is reached.

The decompression is based on block-wise decoding. In intra-coded frames, every block needs to be decompressed with complete inverse wavelet transformation. On the other hand, in predictive frames, only significantly changing blocks are updated so that it can approximate the actual image as accurately as possible while minimizing decompression time. The specific decoding algorithm is as follows. Using the block significance map, we can identify every significant block and its corresponding encoded blocks. For each encoded block, perform the following steps: Read 8bit $b_1^1, ..., b_8^1$ to decide whether one level-0 coefficient and seven level-1 coefficient are zero or not. Next, read 8bit $b_1^2, ..., b_8^2$ to decide whether each eight subblocks has non-zero value or not. If b_k^2, where $k = 1, ..., 8$, is set as 1, read 8bit $c_1^k, ..., c_8^k$ to determine which coefficients of the k-th subblock are non-zero. From the significance maps read above, we can read the actual non-zero coefficients in order. When all values of the coefficients are determined, inverse transformation is performed to get the actual data values and the corresponding block is updated.

Once significant function values are decoded, we perform isosurface extraction. For each significant isovalue, we have a seed set, and hence we can perform the above extraction quickly.

5 Interactive Browsing

While the compression ratio is an important factor for improving I/O performance in the memory and network systems, it is equally important that the visualization browser can read and interactively display compressed streams of multiresolution, time-varying data sets.

5.1 Multi-Resolution Isosurface Rendering

Since an isosurface often contains a lot of triangles, multiresolution techniques are necessary for saving both extraction and rendering

	Extraction time	Triangles#
level2	3110ms	207894
level1	625ms	41624
level0	204ms	11878

Table 1: Isosurface extraction time

time as a tradeoff with visual fidelity. One strength of wavelet transforms is that it provides multiresolution and compressed representations in a consistent format.

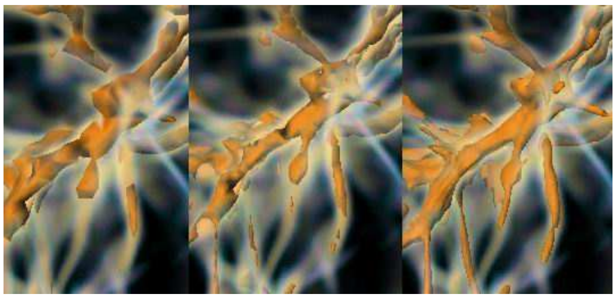

Figure 5: Three different level of an isosurface.

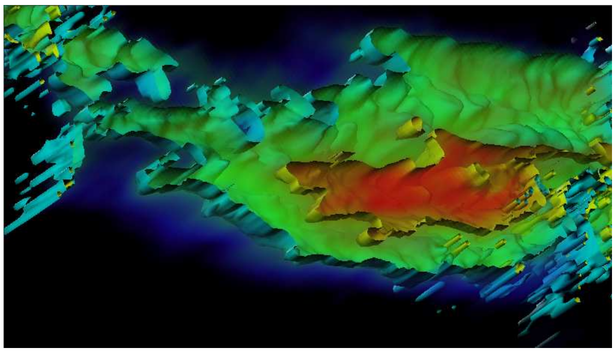

Figure 6: The visualization of isosurfaces and volumes in the oceanographic temperature change data set.

In our block-based wavelet transform, there are 3 levels consisting of one 0-th level, seven 1-st level, and fifty six 2-nd level coefficients. The 0-th level coefficient provides low-pass filtered average value of $4 \times 4 \times 4$ cells. 0-th level and 1-st level coefficients together provide approximated intermediate values averaging $2 \times 2 \times 2$ cells. When fast extraction and rendering of isosurfaces are more important than accuracy, the client can take only the 0-th level and/or 1-st level coefficients, and reconstruct a volume of lower resolution. Since the reconstructed low resolution volume is a good approximation of the original volume, not only is the extracted isosurface a good approximation of original isosurface, but the number of triangles extracted is also reduced. This process has the effect of low-pass filtering the volume, which can remove noise and artifacts incurred by lossy compression. Figure 5 shows three images rendered using the same volumetric data, but different levels of an isosurface.

Data	Res.	type	#frm	1 frm size
Gas	$256 \times 256 \times 256$	float	144	64MB
Ocean	$2160 \times 960 \times 30$	float	115	237MB
Ocean(decim)	$512 \times 256 \times 32$	float	39	16MB

Table 2: Information on time-varying data sets

Compression ratio - 148:1, average RMSE - 0.108

Frame#	Recon. time	RMSE*	Comp. ratio
1(I)	687ms	0.105	40:1
2(P)	177ms	0.131	378:1
3(P)	271ms	0.101	182:1
4(P)	235ms	0.103	234:1

Compression ratio - 301:1, average RMSE - 0.139

Frame#	Recon.time	RMSE*	Comp. ratio
1(I)	489ms	0.127	70:1
2(P)	141ms	0.156	988:1
3(P)	207ms	0.130	422:1
4(P)	188ms	0.134	516:1

Table 3: Compression performance on gas dynamics data set (* : original density range $[0., 8065.299]$)

5.2 Combined Rendering of Isosurface and Volumetric Data

We tested our work on an implementation based on OpenGL. OpenGL gives us the ability to perform depth tests and maintain a depth map. We take advantage of this to render the isosurface and volume in a consistent manner. We take the isosurface as the region we need to see in greater detail, with an opaque or translucent volume in the object space. So we set the isosurface to be completely opaque and render it using OpenGL, with the depth test on. Then, we render the volume from back to front ordering. This is consistent with what is recommended in the OpenGL documentation [OpenGL n. d.]. The rendering result is shown in figure 6.

During the rendering of the isosurface, we build up a depth map, which is used during volume rendering. While isosurfacing in general needs a large amount of time, we already have the seeds required to perform the seed set isocontouring. Hence, we achieve fast isosurface reconstruction times. When we perform volume rendering, to obtain correct transparency results, we render the polygons in a sorted order. While it is generally time consuming to perform polygon sort, Nvidia 3 graphics card's 3d texture mapping capability helps overcome this. The volume data is stored in the graphics card texture memory. We create polygons through a simple incremental algorithm, making slices through the volume. These polygons are rendered with the corresponding texture values in the 3D memory. An interactive transfer function map to control color and opacity values is used to obtain the required images. Depending on the characteristics of the volume's dimensions, we need to adjust the slicing frequency. The oceanographic data set, which is very thin in one dimension, needs a relatively larger number of slices to prevent artifacts and incorrect rendering. If the user navigates through the time varying data, we need to update the slicing direction.

To get slightly better performance, we turn off building the depth buffer (using glDepthMask(0)) when we render the volume, as we are sure of rendering the polygons in order, and since all polygons previously rendered are opaque.

Compression ratio - 183:1, average RMSE - 0.090

Frame#	Recon. time	RMSE**	Comp. ratio
1(I)	124ms	0.076	72:1
2(P)	76ms	0.087	273:1
3(P)	96ms	0.089	226:1
4(P)	86ms	0.091	223:1

Compression ratio - 348:1, average RMSE - 0.177

Frame#	Recon. time	RMSE**	Comp. Ratio
1(I)	106ms	0.157	103:1
2(P)	57ms	0.169	615:1
3(P)	65ms	0.175	493:1
4(P)	64ms	0.178	482:1

Table 4: Compression performance on oceanographic data set (** : original temperature range $[-2.0, 36.0]$)

Scheme#	Comp. ratio	RMSE
FBE	181:1	0.110
FVE	181:1	0.131

Table 5: Comparison of Feature Based Encoding(FBE) and Full Volume Encoding(FVE)

6 Experimental Results

Compression and rendering results were computed on a PC equipped with a Pentium III 800MHz processor, 512MB main memory, and a NVidia GeForce 3 graphics card which has 128MB texture memory. We used standard OpenGL functions for 3D texture-mapping based volume rendering.

Table 2 provides the information about our test data sets. The first data set is generated from a computational cosmology simulation for the formation of large-scale structures in the universe. The data is embedded in a cube 64 megaparsecs(209 million light years) on each side, and models dark matter and gaseous components. This data set has 144 frames. Since the functions in the data set have negligible changes in the last few frames, we have given all our compression results, for this data set, based on the first 100 frames. The second data set is generated by a simulation in the field of oceanography. The original model has an approximate resolution of 1/6 degree (2160 by 960) in latitude and longitude and carries information at 30 depth levels. It includes several time-steps of scalar and vector field data sets like temperature, salinity, velocity, ocean surface height, and ocean depth. The timestep interval is 300 seconds beginning on Feb-16-1991 at 12:00:00. In this paper, we used only the temperature data. Since the original resolution of the data is too high for hardware volume rendering, we decimated it into $512 \times 256 \times 32$ by subsampling and took every third frame.

For testing the performance of our compression scheme, we encoded only high temperature regions ranging between 21.47 and 36.0(celsius degree) in the oceanographic temperature data set and high density regions ranging between 0.23917 and 3.26161 in gas dynamics data set as shown in Figure 7 and 9. After encoding wavelet coefficients, we used *gzip* for lossless compression.

Table 3 and 4 show the reconstruction time, root mean squared error (RMSE), and the compression ratio changes over time in gas dynamics and oceanography data sets. The reconstruction time includes the time for disk read, *gunzip*, and decoding of wavelet coefficients. The RMSE was calculated using only those function values which contributed to the features. The compression ratio was calculated by comparing the size of original time-varying volume data and feature-based compressed data encoded by applying our lossy compression and gzip. As you can see in the tables, the reconstruction time of a P frame is much less than that of an I frame while the compression ratio of P frame is much higher than that of an I frame. The reason for this is that our compression scheme only encodes significantly changing blocks in P frames.

In table 5, we compare our feature based encoding (FBE) scheme with the full volume encoding (FVE) scheme. A rendered image of each scheme is shown in figure 4. Since FBE encodes only the values contributing to the specified features, both the decompression time and the compression ratio are significantly improved with respect to schemes encoding full volumes. While transmission and reconstruction times of volumetric and isosurface features are reduced by FBE encoding, client-side rendering is significantly accelerated by using PC graphics hardware. Timing results of rendering are presented in table 6.

Figure 7 and 8 show a typical frame of the gas dynamics data set compressed with different compression ratios. Figure 9 and 10 show the flow of two isosurfaces of specific temperatures. Figure 8 shows a zoomed view of the same set of volumes rendered in 7. We notice that good visual quality is maintained even at such zoom factors and high compression ratios. While a zoomed image of a volume compressed at a ratio of 148:1 is visually almost the same as the original, we get only a few artifacts at compression ratios of 301:1. Figures 9 and 10 show that similar results were obtained when our compression and rendering scheme was applied to the oceanographic data set. We have used isosurfaces to track the movement of water with a specific temperature and a translucent volumetric region to represent the surrounding temperatures. These figures demonstrate the strength of our scheme in being able to interactively render specific regions of interest with high quality isosurfaces, surrounded by related volumetric data.

Although the wavelet based encoding can generate some losses in volumes as well as the topology in the reconstructed isosurfaces, we can achieve very high compression ratios with acceptable degradation.

Data set	Load	Ext. Time	Tri#	Isosurface	Volume
Gas	701ms	1703ms	135362	312ms	422ms
Ocean	156ms	1640ms	104900	235ms	281ms

Table 6: Timing results of rendering isosurface with amorphous volumetric features in one frame:(the data set name, 3D texture loading time, isosurface extraction time, triangle number of an isosurface, isosurface rendering time, volume rendering time)

7 Conclusion

We described a lossy compression scheme for encoding time-varying isosurfaces with amorphous volumetric features specified by scalar value ranges. Since large time-varying volume data has lots of coherence, compression is necessary for saving storage space, reducing transmission time, and improving the performance of visualizing the time-varying data. From this motivation, we achieved our goals : (i) high compression ratio with minimal image degradation, (ii) fast decompression, by truncating insignificant blocks and wavelet coefficients, and (iii) interactive client-side rendering of compressed isosurfaces and volumetric features. Therefore, our compression scheme is useful for the interactive navigation and exploration of time-varying isosurfaces with amorphous volumetric features residing in local and/or remote data servers. As a future work, we plan to use high order wavelet basis for smooth reconstruction and develop an encoding scheme which provides multi-resolution, time-dependent volumetric representation of interesting features in the compressed format.

Acknowledgements

The authors are grateful to Anthony Thane for allowing us to use his rendering program. We would like to thank Marcelo Alvarez and Prof. Paul Shapiro of the Galaxy Formation and Intergalactic Medium Research Group at UT-Austin for providing the gas density data set, and Prof. Detlef Stammer for providing access to the oceanography simulation data set.

This work was supported in part by NSF grants ACI-9982297, CCR-9988357, a DOE-ASCI grant BD4485-MOID from LLNL/SNL and from grant UCSD 1018140 as part of NSF-NPACI, Interactive Environments Thrust, and a grant from Compaq for the 128 node PC cluster.

References

BAJAJ, C., PASCUCCI, V., AND SCHIKORE, D. R. 1996. Fast isocontouring for improved interactivity. In *Preceedings of the 1996 Symposium for Volume Visualization*, 39–46.

BAJAJ, C., IHM, I., AND PARK, S. 2001. Visualization-specific compression of large volume data. In *Proc. of Pacific Graphics*, 212–222.

BAJAJ, C., IHM, I., AND PARK, S. 2001. 3D RGB image compression for interactive applications. *ACM Transactions on Graphics 20*, 1, 10–38.

CIGNONI, P., MARINO, P., MONTANI, E., PUPPO, E., AND SCOPIGNO, R. 1997. Speeding up isosurface extraction using interval trees. In *IEEE Transactions on Visualization and Computer Graphics*, 158–170.

ELLSWORTH, D., CHIANG, L.-J., AND SHEN, H.-W. 2000. Accelerating time-varying hardware volume rendering using tsp trees and color-based error metrics. In *Proceedings Volume Visualization and Graphics Symposium 2000*, 119–128.

GALL, D. L. 1991. MPEG: A video compression standard for multimedia applications. *Communications of the ACM 34*, 4, 46–58.

GERSTNER, T., AND PAJAROLA, R. 2000. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In *Proceedings Visualization 2000*, T. Ertl, B. Hamann, and A. Varshney, Eds., 259–266.

GUTHE, S., AND STRASER, W. 2001. Real-time decompression and visualization of animated volume data. In *IEEE Visualization 2001*, 349–356.

HOWIE, C., AND BLACK, E. 1994. The mesh propagation algorithm for isosurface construction. *Computer Graphics Forum 13*, 3, 65–74.

IHM, I., AND PARK, S. 1998. Wavelet-based 3d compression scheme for interactive visualization of very large volume data. In *Proceedings of Graphics Interface '98*, 107–116.

KHODAKOVSKY, A., SCHRÖDER, P., AND SWELDENS, W. 2000. Progressive geometry compression. In *Siggraph 2000, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, K. Akeley, Ed., 271–278.

LORENSEN, W., AND CLINE, H. 1987. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM SIGGRAPH '87*, 163–169.

LUM, E. B., MA, K.-L., AND CLYNE, J. 2001. Texture hardware assisted rendering of time-varying volume data. In *IEEE Visualization 2001*, 263–270.

MA, K.-L., AND CAMP, D. M. 2000. High performance visualization of time-varying volume data over a wide-area network status. In *Supercomputing*.

MACHIRAJU, R., ZHU, Z., FRY, B., AND MOORHEAD, R. 1998. Structure-significant representation of structured datasets. *IEEE Transactions on Visualization and Computer Graphics 4*, 2, 117–132.

OPENGL. Developer faq. `http://www.opengl.org/developers/faqs/technical.html`.

PFISTER, H., LORENSEN, B., BAJAJ, C., KINDLMANN, G., SCHROEDER, W., AVILA, L. S., MARTIN, K., MACHIRAJU, R., AND LEE, J. 2001. The transfer function bake-off. *IEEE Transactions on Computer Graphics and Applications 21*, 3, 16–22.

SAID, A., AND PEARLMAN, W. A. 1996. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology 6*, 243–250.

SHAMIR, A., PASCUCCI, V., AND BAJAJ, C. 2000. Multiresolution dynamic meshes with arbitrary deformations. In *Proc of IEEE Visualization Conference 2000*, 423 – 430.

SHAPIRO, J. 1993. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing 41*, 12, 3445–3462.

SHEN, H.-W., HANSEN, C. D., LIVNAT, Y., AND JOHNSON, C. R. 1996. Isosurfacing in span space with utmost efficiency (ISSUE). In *IEEE Visualization '96*, R. Yagel and G. M. Nielson, Eds., 287–294.

SHEN, H.-W. 1998. Isosurface extraction in time-varying fields using a temporal hierarchical index tree. In *IEEE Visualization '98*, 159–166.

SILVER, D., AND WANG, X. 1997. Tracking and visualization turbulent 3d features. *IEEE Transactions on Visualization and Computer Graphics 3*, 2, 129–141.

SUTTON, P. M., AND HANSEN, C. D. 1999. Isosurface extraction in time-varying fields using a temporal branch-on-need tree (T-BON). In *IEEE Visualization '99*, D. Ebert, M. Gross, and B. Hamann, Eds., 147–154.

TAUBIN, G., AND ROSSIGNAC, J. 1998. Geometric compression through topological surgery. *ACM Trans. Gr. 17*, 2, 84–115.

WESTERMANN, R., AND THOMAS, E. 1998. Efficiently using graphics hardware in volume rendering applications. In *In SIGGRAPH '98*, 169–177.

ZHANG, X., BAJAJ, C., AND BLANKE, W. 2001. Scalable isosurface visualization of massive datasets on cots clusters. In *Proceedings of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics, 2001.*, 51–58.

ZHANG, X., BAJAJ, C., AND RAMACHANDRAN, V. 2002. Parallel and out-of-core view-dependent isocontour visualization using random data distribution. In *Joint Eurographics-IEEE TCVG Symposium on Visualization 2002*, 9–18.

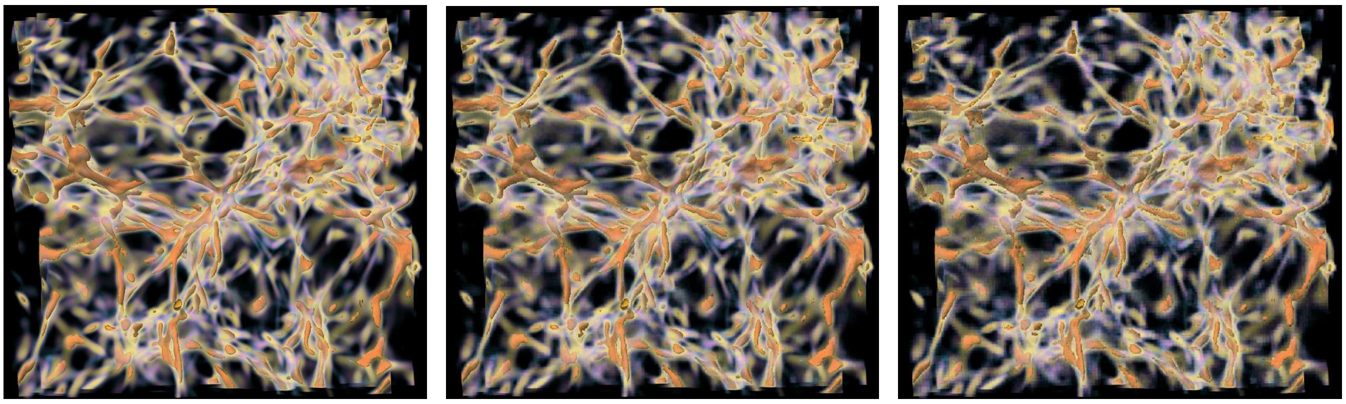

Figure 7: Gas dynamics data set. Original volume (left), compressed volume with compression ratio 148:1 (middle), and compression ratio 301:1 (right).

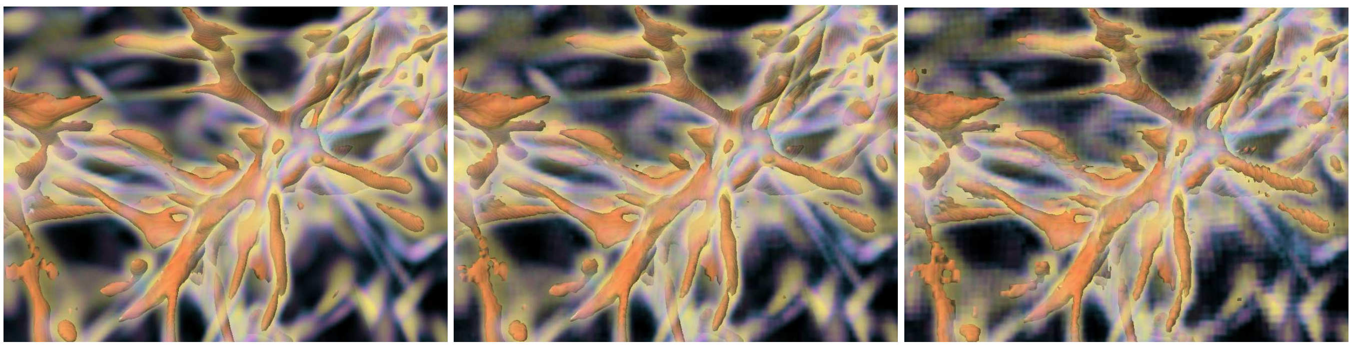

Figure 8: Gas dynamics data set. Zoomed images from figure 7.

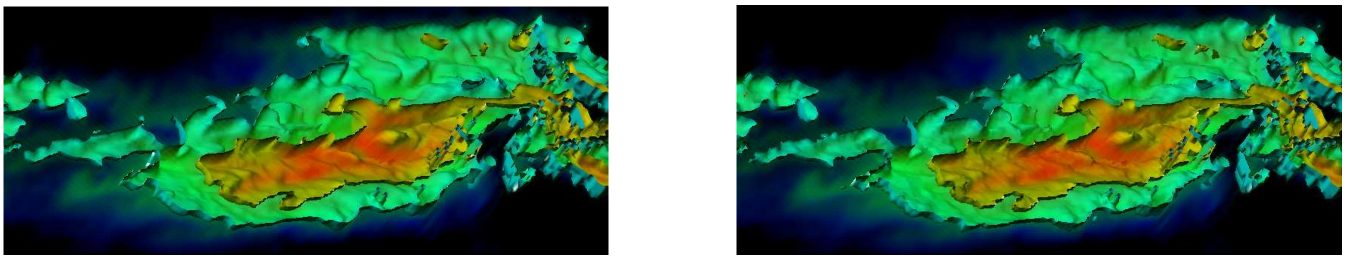

Figure 9: Oceanographic temperature change data set. Original volume (left) and compressed volume with compression ratio 183:1 (right).

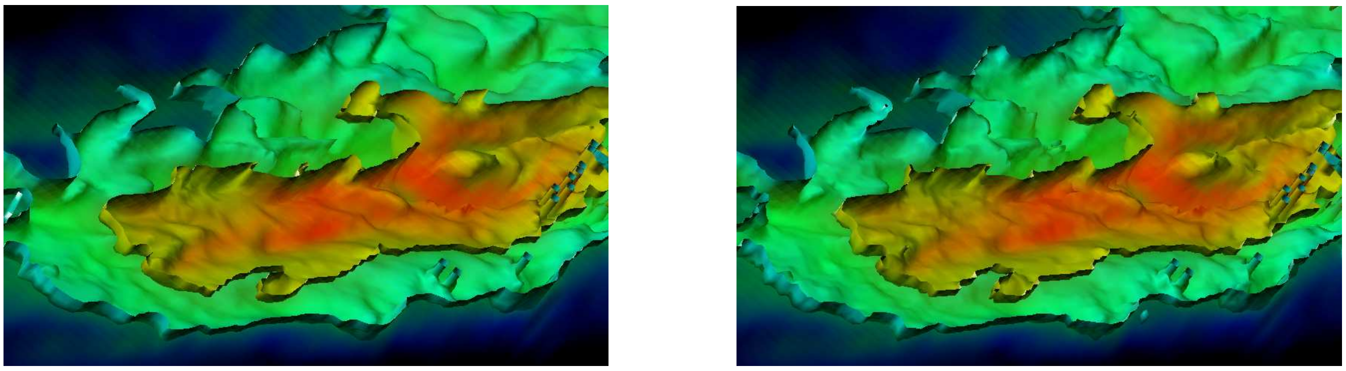

Figure 10: Oceanographic temperature change data set. Zoomed images from figure 9.