Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Vision and Image Understanding 96 (2004) 435–452
Computer Vision and Image Understanding
www.elsevier.com/locate/cviu

Volumetric video compression for interactive playback☆

Bong-Soo Sohn,* Chandrajit Bajaj, and Vinay Siddavanahalli

Department of Computer Sciences, Center for Computational Visualization, Institute for Computational Engineering and Sciences, University of Texas, Austin, TX 78712, USA

Received 1 May 2003; accepted 1 March 2004
Available online 2 July 2004

Abstract

We develop a volumetric video system which supports interactive browsing of compressed time-varying volumetric features (significant isosurfaces and interval volumes). Since the size of even one volumetric frame in a time-varying 3D data set is very large, transmission and on-line reconstruction are the main bottlenecks for interactive remote visualization of time-varying volume and surface data. We describe a compression scheme for encoding time-varying volumetric features in a unified way, which allows for on-line reconstruction and rendering. 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 isosurfaces and interval volumes. The results show that our compression scheme achieves high compression ratio with fast reconstruction, which is effective for interactive client-side rendering of time-varying volumetric features.

Keywords: Time-varying volume visualization; 3D video; Compression; Isocontouring; Hardware-acceleration; Wavelet transform

☆ Visit http://www.ices.utexas.edu/~bongbong/volvideo for accessing video files.

* Corresponding author.
E-mail addresses: bongbong@cs.utexas.edu (B.-S. Sohn), bajaj@cs.utexas.edu (C. Bajaj), skvinay@cs.utexas.edu (V. Siddavanahalli).

1077-3142/$ - see front matter
doi:10.1016/j.cviu.2004.03.010

1. Introduction

Scientific simulations of today are increasingly generating 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 237 MB/frame ($2160 \times 960 \times 30$ float) $\times$ 115 frames, and the gas dynamics data set is 64 MB/frame ($256 \times 256 \times 256$ float) $\times$ 144 frames. To visualize such time-varying volumetric data, *volume rendering* and *isocontouring* techniques are performed frame by frame, so that a user can navigate and explore the data set in space and time. Both rendering techniques have their own strengths. While volume rendering can display amorphous volumetric regions specified by the transfer function with transparency, isocontouring can provide the geometric shape of the surfaces specified by significant isovalues. Both techniques can be combined for better understanding of the overall information present in a volumetric data set. For example, Fig. 1 shows visualization of simulated explosion during galaxy formation through rendering of time-varying volumes and isosurfaces.

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. Efficient data management 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 also be a bottleneck for interactive playback.

From this motivation, we have developed a unified compression scheme for encoding time-varying volumetric features. In most cases, we use the term *feature* to mean an isosurface and/or an interval volume specified by a scalar value range. An interval volume can be a whole volume. The inputs for compression 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 in compression is to

- compress time-varying isosurfaces and interval volumes in a unified way;
- reduce the size of time-varying volumetric data with minimal image degradation;

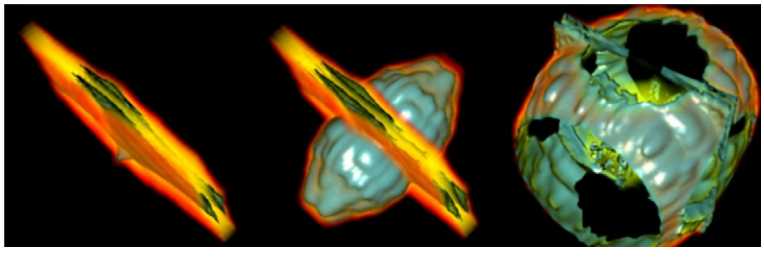

Fig. 1. Interactive volumetric video playback of simulated explosion gas dynamics.

- allow real-time reconstruction and rendering with PC graphics hardware acceleration.

We 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 the compression of time-varying volumetric features 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 volumetric frame 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 their previous frames. 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 in isosurfaces and volumes, and their evolution by using the feature tracking method [24]. The reconstructed features can be rendered in real-time using PC graphics hardware.

Fig. 2 shows the overall architecture of our interactive volumetric video framework. The algorithm requires the features to be defined before compression. These features can be identified manually or by automatic feature detection tools [18]. 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.

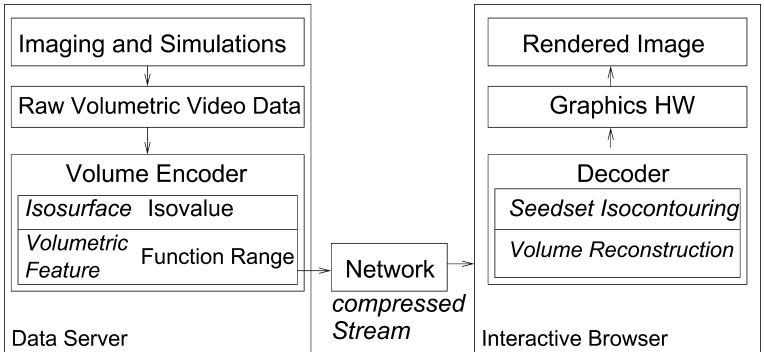

Fig. 2. Interactive volumetric video: remote streaming and display pipeline.

The remainder of this paper is organized as follows. First, the related work is described. Then, in Section 3, an architecture for displaying volumetric video is described. In Section 4, a 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.

2.1. 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 Khodakovsky et al., Taubin and Rossignac [12,27]. A compression scheme specialized for isosurfaces [29] 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 image compression techniques are geared towards achieving the best compression ratio with minimal distortion in the reconstructed images. JPEG and MPEG [6] are developed for compressing still images and 2D video data with controllable size and distortion trade-off. Embedded coding algorithms such as

embedded zero tree wavelet (EZW) [21] and set partitioning in hierarchical trees (SPIHT) [19] are useful for progressive transmission and multimedia applications.

For 3D image compression, Ihm and Park [11] described a wavelet-based 3D compression scheme for the visible human data and later extended it to 3D RGB image compression for interactive applications such as light-field rendering and 3D texture mapping [1]. Compression ratios can be improved by capturing and encoding only significant structures and features in the data set [2,16]. In those compression schemes, the primary goal is fast random access to data, while maintaining high compression ratios. This allows interactive rendering of large volume data sets [9].

Guthe and Straser [8] 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. [14] exploit texture hardware for both rendering and decompression. Since data are transferred in the compressed format between different memory systems, I/O time is significantly reduced.

2.2. Time-based data structures

Time-space partitioning (TSP) tree was introduced and accelerated later by using 3D texture mapping hardware [5] 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 [22] proposed the temporal hierarchical index (THI) tree data structure for single resolution isocontouring of time-varying data, by an extension of his ISSUE algorithm [23]. The THI tree provides a compact search structure, while retaining optimal search time. Hence, expensive disk operations for retrieving search structures are reduced. Sutton and Hansen [26] 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. [20] developed an adaptive multiresolution data structure for time dependent polygonal meshes called time-direct acyclic graph (T-DAG). 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.

2.3. High performance visualization systems

Ma and Camp [15] describes a remote visualization system under the wide area network environment for visualization of time-varying data sets. Current state-of-the-art graphics hardware enables real-time volume and isosurface rendering [28] and decompression [14].

2.4. 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 [13] visits each cell

in a volume and performs appropriate local triangulation for generating the isosurface. To avoid visiting unnecessary cells, accelerated algorithms [4] minimizing the time to search for contributing cells are developed.

The contour propagation algorithm is used for efficient isosurface extraction [3,10]. 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 [7] and view-dependent techniques [30] are useful to reduce the number of triangles in an isosurface.

3. Interactive volumetric video

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 read each frame from the data server and render 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 multiresolution, time-dependent volumetric representation of interesting features (isosurface, volume within a range) of the data, generated and stored in a compressed format. The intermediate multiresolution representation permits trade-offs 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.

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 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 $V = \{\{I_1, P_{11}, P_{12}, ..., P_{1_{p1}}\}, ..., \{I_N, P_{N1}, P_{N2}, ..., P_{N_{pN}}\}\}$ where I_i is an intra-coded frame in the ith temporal group and P_{ij} is the jth predictive frame in the ith 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. 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 Fig. 3. 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 kth 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 3D 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

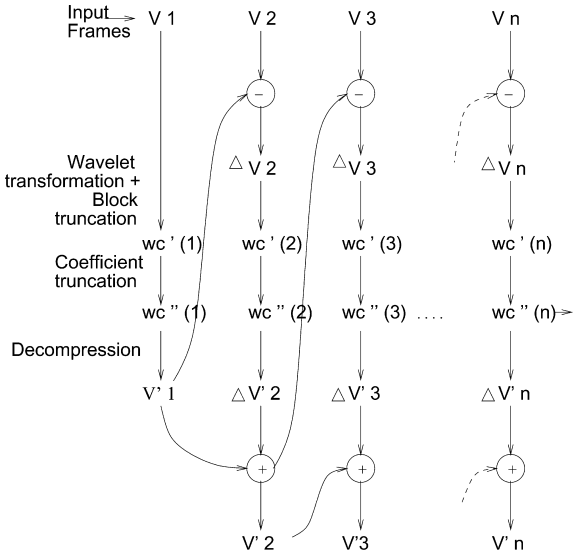

Fig. 3. Overall compression algorithm.

the ith 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 ith 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 ith feature is less than τ_i, the coefficient is truncated into zero.

(6) *Encoding:* The overall encoding scheme is shown in Fig. 4. 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,

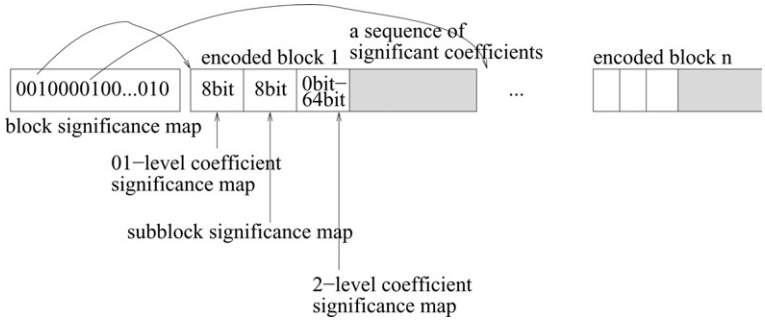

Fig. 4. Suggested encoding scheme for supporting fast decompression and high compression ratios.

resulting in a sequence of encoded blocks. We classify 64 coefficients in a block as one level-0 coefficient, seven 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 a sequence. Then, for each significant blocks, we store an 8 bit map representing whether the one level-0 and seven level-1 coefficients are zero or not. Next, we have another 8 bit map representing whether each eight $2 \times 2 \times 2$ subblock has non-zero wavelet coefficients followed by a significance map for representing non-zero level-2 coefficients. After storing the level-2 coefficient significance maps, the actual values of non-zero wavelet coefficients are stored in order. We used two bytes for storing each coefficient value. Lossless compression is further applied to improve compression.

4.2. Seed cells insertion

To allow browsing clients to quickly extract 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 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.

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, $\Delta V'_k = \textit{inverse transformation}\ (W\Delta V_k)$ 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 8 bit $b_1^1, \ldots, b_8^1$ to decide whether one level-0 coefficient and seven level-1 coefficient are zero or not. Next, read 8 bit $b_1^2, \ldots, b_8^2$ to decide whether each eight subblocks has non-zero value or not. If b_k^2, where $k = 1, \ldots, 8$, is set as 1, read 8 bit $c_1^k, \ldots, c_8^k$ to determine which coefficients of the kth subblock are non-zero. From the significance maps read

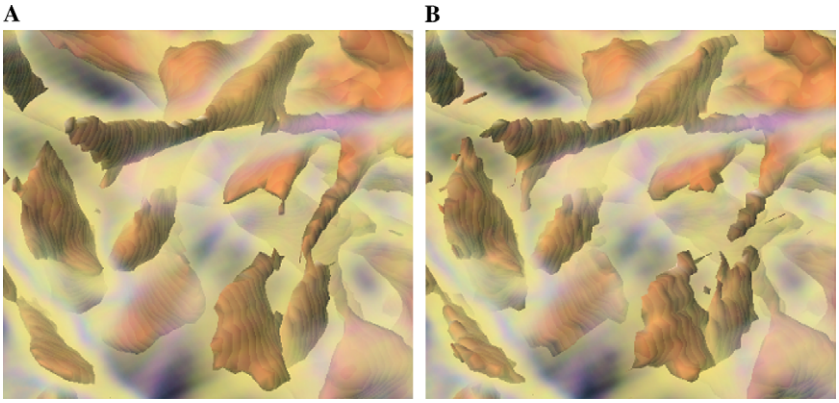

Fig. 5. Performance comparison of different encoding schemes. Although the compression ratio of (A) and (B) is same (181:1), the quality of the reconstructed image using (A) is better than (B). RMSE of (A) is 0.110 and (B) is 0.131. (A) Only the values contributing to features are encoded. (B) Every value in the volume is encoded.

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 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. Multiresolution isosurface rendering

Since an isosurface often contains a lot of triangles, multiresolution techniques are necessary for saving both extraction and rendering time as a trade-off with visual fidelity. One strength of wavelet transforms is that it provides multiresolution and compressed representations in a consistent format.

In our block-based wavelet transform, there are three levels consisting of one 0th level, seven 1st level, and fifty six 2nd level coefficients. The 0th level coefficient provides low-pass filtered average value of $4 \times 4 \times 4$ cells. 0th level and 1st 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 0th level and/or 1st 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. Fig. 6 shows three images rendered using the same volumetric data, but different levels of an isosurface.

5.2. Combined rendering of isosurface and volumetric data

We combine isosurface and volume rendering to take advantages of both techniques. We take an isosurface as the region we need to see in greater detail, with an opaque or translucent volume in the object space. The rendering results are shown and compared in Fig. 7. In this data set, isosurface extraction provides a shaded surface representing a specific temperature value. Volume rendering shows temperatures

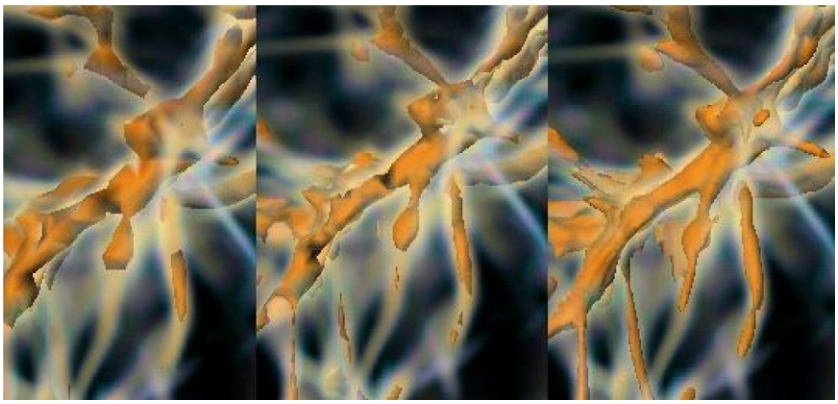

Fig. 6. Three different levels of an isosurface. The number of isosurface triangles and extraction time for level 0 (left), level 1 (middle), and level 2 (right) are (11,878, 204 ms), (41,624, 625 ms), and (20,7894, 3110 ms).

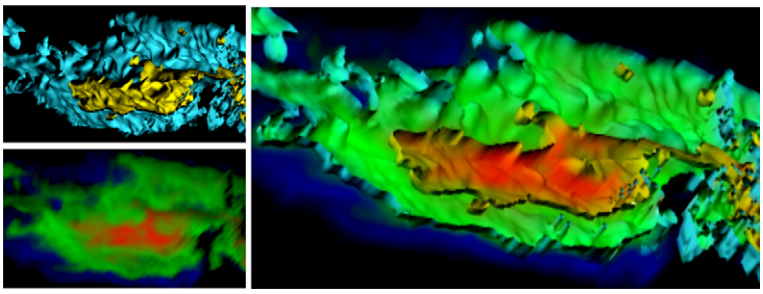

Fig. 7. The visualization of the oceanographic temperature change data set. Upperleft: isosurface rendering, Lowerleft: volume rendering, right: combined rendering of isosurface and volume.

around the isosurface. By combining both techniques, visual information contained in the rendering is enhanced.

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. So we set the isosurface to be completely opaque and render it using OpenGL, with the depth test on. Then, we render a set of 3D textured polygons sliced from a volume from back to front ordering. This is consistent with what is recommended in the OpenGL documentation [17].

During the rendering of the isosurface, we build up a depth map, which is used during volume rendering. While isocontouring 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. 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. If the user rotates a volume, 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.

6. Experimental results

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

Table 1 provides 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. 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 an oceanographic simulation and represents temperature changes in the ocean over time. The original

Table 1
Information on time-varying data sets

Data	Res.	Type	No. of frames	1 frame size (MB)
Gas	$256 \times 256 \times 256$	Float	144	64
Ocean(decim)	$512 \times 256 \times 32$	Float	39	16
Hemoglobin	$128 \times 128 \times 128$	Float	30	8

model has an approximate resolution of 1/6 degree (2160×960) in latitude and longitude and carries information at 30 depth levels. 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. The third data set is obtained from hemoglobin dynamics simulation. The simulation generated a sequence of hemoglobin pdb files with 30 time steps and electron density volumes are computed from each pdb file.

For testing the performance of our compression scheme, we encoded only high temperature regions ranging between 21.47 and 36.0 °C 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 Figs. 8 and 10. After encoding wavelet coefficients, we used *gzip* for lossless compression.

Tables 2 and 3 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 Fig. 5, we compare our feature based encoding (FBE) scheme with the full volume encoding (FVE) scheme. 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.

Table 2
Compression performance on gas dynamics data set

Frame#	Recon. time (ms)	RMSE[a]	Comp. ratio
Average compression ratio — 148:1, RMSE — 0.108			
1(I)	687	0.105	40:1
2(P)	177	0.131	378:1
3(P)	271	0.101	182:1
4(P)	235	0.103	234:1
Average compression ratio — 301:1, RMSE — 0.139			
1(I)	489	0.127	70:1
2(P)	141	0.156	988:1
3(P)	207	0.130	422:1
4(P)	188	0.134	516:1

[a] Original density range [0, 8065.299].

Table 3
Compression performance on oceanographic data set

Frame#	Recon. time (ms)	RMSE[a]	Comp. ratio
Average compression ratio — 183:1, RMSE — 0.090			
1(I)	124	0.076	72:1
2(P)	76	0.087	273:1
3(P)	96	0.089	226:1
4(P)	86	0.091	223:1
Average compression ratio — 348:1, RMSE — 0.177			
1(I)	106	0.157	103:1
2(P)	57	0.169	615:1
3(P)	65	0.175	493:1
4(P)	64	0.178	482:1

[a] Original temperature range [−2.0, 36.0].

Figs. 8 and 9 show a typical frame of the gas dynamics data set compressed with different compression ratios. Figs. 10 and 11 show the flow of two isosurfaces of specific temperatures. Fig. 9 shows a zoomed view of the same set of volumes rendered in Fig. 8. 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

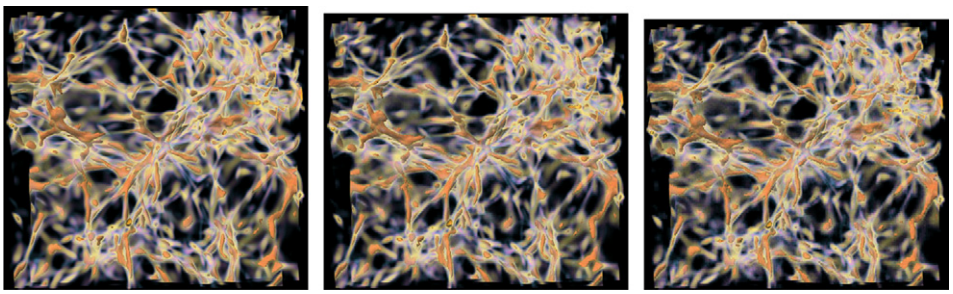

Fig. 8. Gas dynamics data set. Original volume (left), reconstructed volume with compression ratio 148:1 (middle), and compression ratio 301:1 (right).

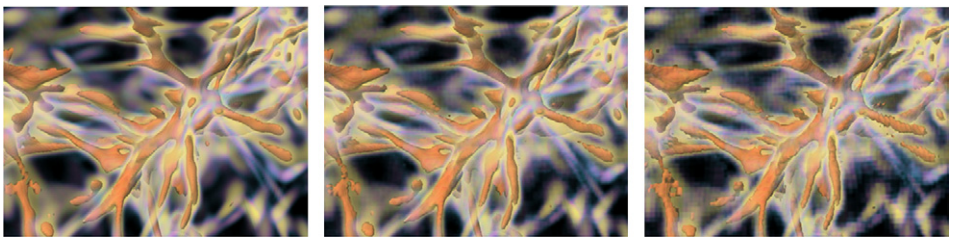

Fig. 9. Gas dynamics data set. Zoomed images from Fig. 8.

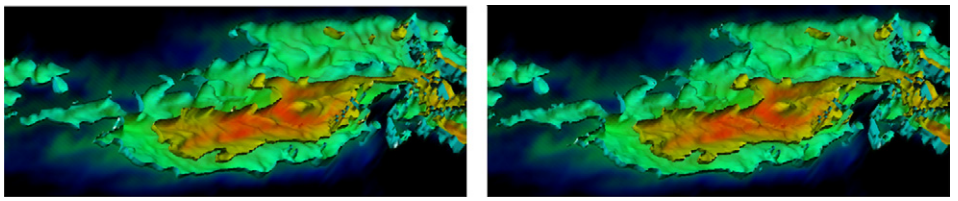

Fig. 10. Oceanographic temperature change data set. Original volume (left) and reconstructed volume with compression ratio 183:1 (right).

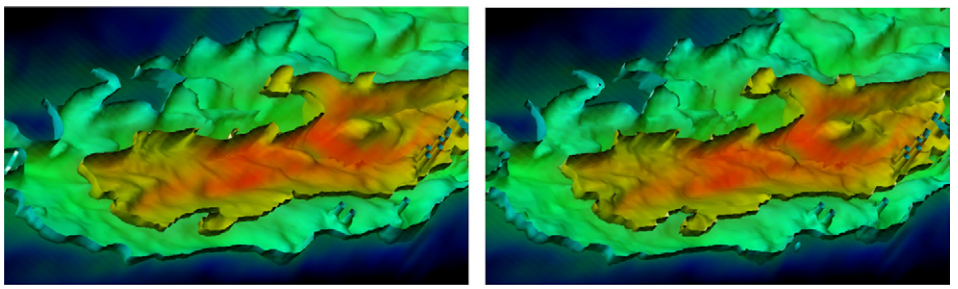

Fig. 11. Oceanographic temperature change data set. Zoomed images from Fig. 10.

at compression ratios of 301:1. Figs. 10 and 11 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. Timing results of rendering are presented in Table 4.

Fig. 12 shows the compression result of hemoglobin dynamics simulation. Volume rendering was applied to each compressed volumetric frame and we measured the frame rate. We achieved high compression ratio (110:1) and high frame rate (6.3 frame/s) with reasonable reconstruction quality.

Generally, the frame rate of volumetric video playback is mostly dependent on the resolution of volumes and the number of triangles in the extracted isosurfaces. The size of rendering is also an important factor because the texture based volume rendering relies on per pixel operations.

Table 4
Timing results of rendering isosurfaces with amorphous volumetric features in one frame: (the data set name, 3D texture loading time, isosurface extraction time, number of triangles in the isosurfaces, isosurface rendering time, and volume rendering time)

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

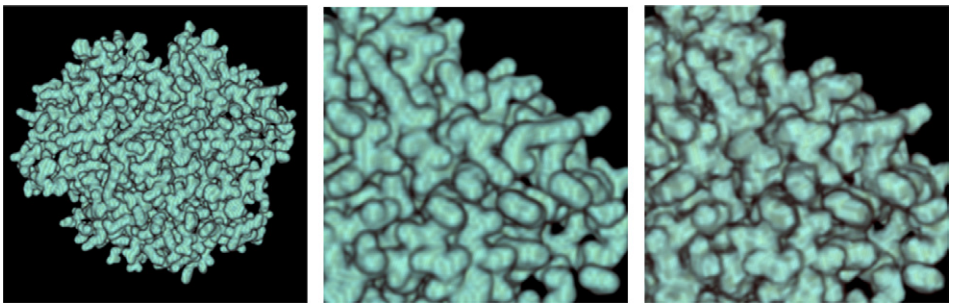

Fig. 12. Hemoglobin dynamics simulation. Left: original volume, Middle: zoomed original volume, Right: zoomed reconstructed volume with compression ratio 110:1.

Although 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.

7. Conclusion

We describe a lossy compression scheme for encoding time-varying isosurfaces with amorphous volumetric features specified by scalar value ranges. Since large time-varying volume data have significant coherence, compression is necessary for saving storage space, reducing transmission time, and improving the performance of visualizing the time-varying data. We have achieved several of 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 multiresolution 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.

Acknowledgments

An early version of this paper appeared in VolVis02 [25]. We are grateful to Anthony Thane for developing the CCV volume rendering tool (Volume Rover) and library, and Xiaoyu Zhang and John Wiggins for writing the volumetric molecular blurring code. The cosmological simulations were performed by Hugo Martel, Paul R. Shapiro, and Marcelo Alvarez, of the Galaxy Formation and Intergalactic Medium Research Group at UT-Austin. The oceanographics simulation data were provided by Professor Detlef Stammer, Physical Oceanography Research Division of the Scripps Institution of Oceanography, under the National Partnership for Advanced Computing Infrastructures project. The hemoglobin dynamics (pdb files) were provided by David Goodsell of Scripps Research Institute.

This work was supported in part by NSF Grants ACI-9982297, CCR-9988357, ACI-0220037, a DOE-ASCI Grant BD4485-MOID from LLNL/SNL and from Grant UCSD 1018140 as part of NSF-NPACI, Interaction Environments Thrust, and a grant from Compaq for the 128 node PC cluster.

References

[1] C. Bajaj, I. Ihm, S. Park, 3D RGB image compression for interactive applications, ACM T. Graphic. 20 (1) (2001) 10–38.
[2] C. Bajaj, I. Ihm, S. Park, Visualization-specific compression of large volume data, in: Proc. Pacific Graphics, Tokyo, Japan, 2001 pp. 212–222.
[3] C. Bajaj, V. Pascucci, D. R. Schikore, Fast isocontouring for improved interactivity, in: Proc. of the 1996 Symp. for Volume Visualization, 1996 pp. 39–46.
[4] P. Cignoni, P. Marino, E. Montani, E. Puppo, R. Scopigno, Speeding up isosurface extraction using interval trees, IEEE Trans. Vis. Comput. Gr. (1997) 158–170.
[5] D. Ellsworth, L.-J. Chiang, H.-W. Shen, Accelerating time-varying hardware volume rendering using tsp trees and color-based error metrics, in: Proc. Volume Visualization and Graphics Symposium 2000, 2000 pp. 119–128,.
[6] D. Gall, MPEG: a video compression standard for multimedia applications, Commun. ACM 34 (4) (1991) 46–58.
[7] T. Gerstner, R. Pajarola, Topology preserving and controlled topology simplifying multiresolution isosurface extraction, in: T. Ertl, B. Hamann, A. Varshney, (Eds.), Proc. Visualization 2000, 2000, pp. 259–266.
[8] S. Guthe, W. Straser, Real-time decompression and visualization of animated volume data, in: IEEE Visualization 2001, 2001, pp. 349–356.
[9] S. Guthe, M. Wand, J. Gonser, W. Straßer, Interactive rendering of large volume data sets, in: Proc. IEEE Visualization Conference, 2002, pp. 54–60.
[10] C.T. Howie, E.H. Black, The mesh propagation algorithm for isosurface construction, Comput. Graph. Forum 13 (3) (1994) 65–74.
[11] I. Ihm, S. Park, Wavelet-based 3d compression scheme for interactive visualization of very large volume data, in: Proc. Graphics Interface '98, 1998, pp. 107–116.
[12] A. Khodakovsky, P. Schröder, W. Sweldens, Progressive geometry compression, in: K. Akeley (Ed.), Siggraph 2000, Computer Graphics Proceedings, ACM Press, ACM SIGGRAPH, Addison Wesley Longman, 2000, pp. 271–278.
[13] W. E. Lorensen, H. E. Cline, Marching cubes: a high resolution 3d surface construction algorithm, in: ACM SIGGRAPH '87, 1987, pp. 163–169.
[14] E. B. Lum, K.-L. Ma, J. Clyne, Texture hardware assisted rendering of time-varying volume data, in: IEEE Visualization 2001, 2001, pp. 263–270.
[15] K.-L. Ma, D. M. Camp, High performance visualization of time-varying volume data over a wide-area network status, in: Supercomputing, 2000.
[16] R. Machiraju, Z. Zhu, B. Fry, R. Moorhead, Structure-significant representation of structured datasets, IEEE Trans. Vis. Comput. Gr. 4 (2) (1998) 117–132.
[17] OpenGL. Developer faq. http://www.opengl.org/developers/faqs/technical.html..
[18] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L. Sobierajski Avila, K. Martin, R. Machiraju, J. Lee, The transfer function bake-off, IEEE Trans. Comput. Graph. Appl. 21 (3) (2001) 16–22.
[19] A. Said, W.A. Pearlman, A new fast and efficient image codec based on set partitioning in hierarchical trees, IEEE Trans. Circ. Syst. Vid. 6 (1996) 243–250.
[20] A. Shamir, V. Pascucci, C. Bajaj, Multi-resolution dynamic meshes with arbitrary deformations, in: Proc. of IEEE Visualization Conference 2000, Salt Lake City, Utah, 2000 pp. 423–430.
[21] J. Shapiro, Embedded image coding using zerotrees of wavelet coefficients, IEEE Trans. Signal Process 41 (12) (1993) 3445–3462.

[22] H.-W. Shen, Isosurface extraction in time-varying fields using a temporal hierarchical index tree, in: IEEE Visualization '98, 1998 pp. 159–166.
[23] H.-W. Shen, C.D. Hansen, Y. Livnat, C.R. Johnson, Isosurfacing in span space with utmost efficiency (ISSUE), in: R. Yagel, G. M. Nielson, (Eds.), IEEE Visualization '96, 1996, pp. 287–294.
[24] D. Silver, X. Wang, Tracking and visualization turbulent 3d features, IEEE Trans. Vis. Comput. Gr 3 (2) (1997) 129–141.
[25] B.-S. Sohn, C. Bajaj, V. Siddavanahalli, Feature based volumetric video compression for interactive playback, in: IEEE/SIGGRAPH Symp. on Volume Visualization and Graphics 2002, 2002, pp. 89–96.
[26] P. M. Sutton, C.D. Hansen, Isosurface extraction in time-varying fields using a temporal branch-on-need tree (T-BON). in: D. Ebert, M. Gross, B. Hamann, (Eds.), IEEE Visualization '99, San Francisco, 1999 pp. 147–154.
[27] G. Taubin, J. Rossignac, Geometric compression through topological surgery, ACM T. Graphic. 17 (2) (1998) 84–115.
[28] R. Westermann, T. Ert, Efficiently using graphics hardware in volume rendering applications, in: SIGGRAPH '98, 1998 pp. 169–177.
[29] X. Zhang, C. Bajaj, W. Blanke, Scalable isosurface visualization of massive datasets on cots clusters, in: Proc. IEEE Symp. on Parallel and Large-Data Visualization and Graphics 2001, 2001, pp. 51–58.
[30] X. Zhang, C. Bajaj, and V. Ramachandran, Parallel and out-of-core view-dependent isocontour visualization using random data distribution, in: Joint Eurographics-IEEE TCVG Symposium on Visualization 2002, 2002 pp. 9–18.