Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2002), pp. 1–6
D. Ebert, P. Brunet, I. Navazo (Editors)

Volumetric Video Compression for Real-Time Playback

Bong-Soo Sohn , Chandrajit Bajaj , Sanghun Park and Vinay K. Siddavanahalli

Department of Computer Sciences & TICAM
The University of Texas at Austin
Austin, TX, USA

Abstract

In this paper, we describe a compression scheme for volumetric video data(3D space $\times$ 1D time) where each frame of the volume is decompressed and rendered in real-time. Since even one frame size of volume is very large, run-time decompression can be a bottleneck for real-time playback of time-varying volume data. To increase the run-time decompression speed and compression ratio, we decompose the volume into small blocks and only update significantly changing blocks. The results show that our compression scheme compromises decompression speed and image quality well enough for interactive time-varying visualization.

1. Introduction

As computing power and scanning precision rapidly increases, scientific simulation or measurements generate more and more densely sampled time-varying 3D volume data which has a very large size. For example, the size of oceanography temperature data tested in this paper is about 240MB/frame ($2160 \times 960 \times 30$ float type) $\times$ 115 frames.

While current state-of-the-art graphics hardware allows very fast volume rendering, transfer of such large data between disk and main memory or between main memory and graphics hardware memory on demand can become a bottleneck due to the limited bandwidth of memory systems. Therefore, efficient data management scheme is an important factor in the rendering performance. To reduce the size of 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 bottleneck for real-time playback.

From this motivation, we developed an efficient compression scheme for real-time playback of time-varying volume data. The input is discretized time-varying volume data V, which can be represented as $V = \{V_1, V_2, \ldots, V_T\}$, where $V_t = \{f^t_{i,j,k} | i, j, k$ are indices of x, y, z coordinate$\}$ is the volume at time step t. Our primary goal in compression is

- to reduce the size of time-varying volume data set with minimal image distortion.
- to allow fast run-time decompression.
- to make decompression suitable for real-time hardware volume rendering.

We are going to borrow the idea of MPEG compression to efficiently exploit spatial and temporal coherence in the data set. However, direct extension of MPEG for 2D video compression to 3D video compression is not suitable for our goal because decompression of every blocks in each frame of large volume data is not fast enough for interactive visualization.

Wavelet transformation is widely used for 2D and/or 3D image compression. By truncating insignificant coefficients after wavelet transformation, it achieves high compression ratio while keeping minimal image distortion. However, complete inverse transformation for each frame is too time-consuming for interactive visualization of large data. Since there are only a few changes between consecutive volume frames, transformation and inverse transformation of only the significantly changing part instead of the full frame can improve 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 high compression ratios and fast decompression speeds.

In addition to efficient compression, fast rendering of volume frames is important for real-time playback of 3D video. Using graphics hardware based on 3D texture mapping, we implemented real-time volume rendering of large data.

Figure 1 shows the overall architecture of our real-time volumetric video display system. To save disk storage space and overcome the limitation of I/O bandwidth in memory systems, a series of compressed frames are read from disks. Once each compressed frame is read, it is decompressed in software and the reconstructed image array is sent to the texture memory in the graphics hardware for displaying. This architecture can allow users to explore and interact with the volume data in space and time, which is our ultimate goal.

The remainder of this paper is organized as follows. In the next section, related work is described. In section 3, volumetric video displaying architecture is described. In section 4, volumetric video compression scheme supporting real-time decompression is proposed. In section 5, experimental results are described. Finally, in section 6, we give a conclusion.

2. Related Work

Visualization of time-varying volume data has been a challenging problem due to the overwhelming data size. Hierarchical data structures, high performance visualization system and compression have been introduced to deal with such large data.

Time-Space Partitioning(TSP) Tree [6] is introduced and accelerated later by using 3D texture mapping hardware [9] for fast volume rendering of time-varying fields. 3D space is partitioned using octree and each octree node has a binary time tree bisecting time recursively such that each binary tree node has both spatial and temporal information. Spatial and temporal variance metrics are retained in each node and used to skip insignificant rendering operations, or to cache the rendered image for later reuse when the viewpoint is fixed. This approach significantly improves volume rendering speed and reduces I/O overhead by exploiting both spatial and temporal coherence.

Ma and Camp [7] describes remote visualization system under the wide area network environment for the visualization of time-varying dataset.

Since scientific data tend to be very large and have lots of redundancy, people prefer to use compressed data for efficient use of memory and I/O bandwidth. There are several image compression techniques, most of which are geared towards achieving the best compression ratio with minimal distortion in the reconstructed images. JPEG[3] and MPEG[2] are developed for compressing still images and 2D video data with controllable size and distortion tradeoff. Sengupta et al. [5] proposes fast video coding using wavelet transformation of differences between consecutive images. Recently, Ihm et al.[10] developed a 3D static image compression scheme which successfully compromises random reconstruction time and compression ratio for interactive applications. Ma et al. [8] used compression of time-varying volume data for efficient volume rendering. Guthe et al. [4] applied the MPEG algorithm to time-varying volume data using wavelet transformation. They compare the effects of motion compensation and using various wavelet basis function. Lum et al. [1] exploits texture hardware for both rendering and decompression. Since data is transfered in the compressed format between different memory systems, I/O time is significantly saved. However, high compression ratio is not expected due to the simple compression scheme.

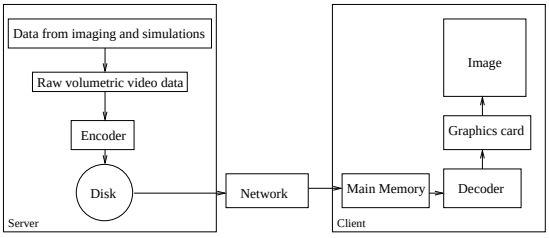

Figure 1: *3D video displaying process*

3. Volumetric Video

Like widely used 2D video, volumetric video displays a sequence of 3D images over time, frame by frame. While in 2D video, user can only look at continually updated 2D images in a passive way, volumetric video, or time-varying volume visualization system allows users to explore and navigate the 3D data both in space and time. Considering that most of the scientific simulation generates dynamic 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 disk and rendering the volume with given visualization parameters. Since most of time-varying scientific data are very large and have high spatial and temporal coherence, it is natural to apply compression for reducing storage overhead. In addition, one frame size can be too large to read from disk in real-time and hence requires efficient compression scheme for reducing the I/O bandwidth limitation. 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 3D video because they usually decompose an image into blocks and decode every blocks during decompression. To reduce the run-time decompression, we limited the number of blocks to decode by ordering the blocks based on their significance and encode only significantly changing blocks.

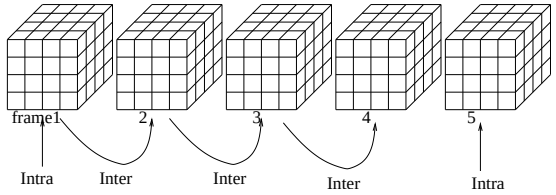

Figure 2: *Intra-coded and predictive frame classification*

4. Compression Scheme

4.1. Compression

The input dataset to be compressed is time dependent volume data, $V = \{V_1, V_2, ..., V_N\}$. Each frame of the volume is classified as either an intra-coded frame or a predictive frame as shown in figure 2. The whole data set is represented as $V = \{\{I_1, P_{11}, P_{12}, ..., P_{1k_1}\}, ..., \{I_n, P_{n1}, P_{n2}, ..., P_{nk_n}\}\}$ where I_i is a intra-coded frame in $i-th$ temporal group and P_{ij} is a predictive frames in $i-th$ temporal group.

Assuming that there are only small changes between consecutive frames, wavelet transformation of changes instead of the whole frame yields higher compression ratios and faster decompression times. Therefore, compression of intraframe is independent of other frames while compression of predictive frames are dependent on previous frames in the same temporal group. Overall compression algorithm is as follows. Note that compression is performed on each volume frame by frame. All 3D frames are decomposed into $4 \times 4 \times 4$ blocks and wavelet transformation is performed on each block.

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 compressed V_{k-1}. If V_k is an intra-coded frame, we assume V'_{k-1}=0.
2. **wavelet transformation** : ΔWV_k = wavelet transformation of (ΔV_k). Compute coefficients $c_1, ..., c_m$ representing ΔV_k in a normalized three-dimensional Haar wavelet basis.
3. **sorting and truncation of wavelet coefficients** : Sort normalized wavelet coefficients in the decreasing order to produce the coefficients sequence $c_{\pi(1)}, c_{\pi(2)}, ..., c_{\pi(m)}$. Given the ratio of non-zero coefficients τ, set coefficients $c_{\pi(\hat{m})}$ as zero where $\hat{m} = m \times \tau$.
4. **sorting and truncation of blocks** : Block truncation is performed on only predictive frames. Each block has 64 wavelet coefficients. We sort the blocks by comparing the sum of coefficients square in the decreasing order to produce the block sequence $b_{\pi'(1)}, b_{\pi'(2)}, ..., b_{\pi'(n)}$. Given the ratio of non-zero blocks, λ, set blocks $b_{\pi'(\hat{n})}$ as zero where $\hat{n} = n \times \lambda$. Here, zero block means that all 64 wavelet coefficients of the block are zero.
5. **encoding** : Overall encoding scheme is shown in figure 3. The encoding is performed on each block, resulting in a sequence of encoded blocks. We classify 64 coefficients in a block as one 0-level coefficient, 7 1-level coefficients and 8×7 2-level coefficients to take advantage of 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 0-level and seven 1-level 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 2-level coefficients. After storing 2-level coefficient significance maps, actual two byte values of non wavelet coefficients are stored in order.

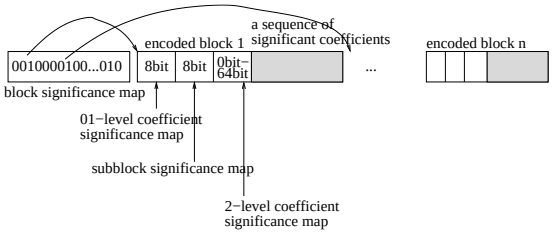

Figure 3: *suggested encoding scheme for supporting fast decompression and high compression ratio*

4.2. Run-Time Decompression

Since we have a sequence of wavelet encoded volume ΔWV_k, we can get an approximated image V_k by decoding and performing an inverse wavelet transformation. More specifically, $\Delta V_k = inverse\,transformation(\Delta WV_k)$ 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 ΔV_i with no dependency on other frames. Once V_i is reconstructed, succeeding predictive frames can be decoded frame by frame until 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 close as possible while minimizing decompression time. The specific decoding algorithm is as follows. Using block significance map, we can identify every significant block and its corresponding encoded blocks. For each encoded block, perform following steps. Read 8bit $b_1^1, ..., b_8^1$ to decide whether one 0-level coefficient and seven 1-level 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

Data	Res.	type	#frm	1frm size
Gas	$256 \times 256 \times 256$	float	144	64MB
Ocean	$2160 \times 960 \times 30$	float	115	240M

Table 1: *Information on Time-Varying Data Set*

τ, λ	comp. ratio	decomp. time(sec)(I,P)
$\tau = 10\%, \lambda = 10\%$	40 : 1	2.4 , 0.27
$\tau = 10\%, \lambda = 1\%$	93 : 1	2.3 , 0.04
$\tau = 5\%, \lambda = 10\%$	66 : 1	2.16 , 0.24
$\tau = 5\%, \lambda = 1\%$	148 : 1	2.17 , 0.032

Table 2: *test results on universe gas density data*

b_k^2, where $k = 1,...,8$, is set on, read 8bit $c_1^k,...,c_8^k$ to determine which coefficients of k-th subblock are non-zero. From significance maps read above, we can read actual non-zero coefficients in order. When all values of coefficients are determined, inverse transformation is performed to get actual data values and corresponding block is updated.

5. Experimental Results

Compression and decompression results were computed on a Silicon Graphics ONYX2 InfiniteReality2 system with 24 R12000 processors, 25GB main memory. We used OpenGL and OpenGL Volumizer for 3D texture-mapping based volume rendering which is capable of real-time volume rendering.

Table 1 provides the information about our test data set.

We tested compression ratio, average frame decompression time(intra-coded frame I, predictive P) for various sets of $\tau = 1\%, 5\%, 10\%$ (the ratio of non-zero wavelet coefficients) and $\lambda = 1\%, 5\%, 10\%$ (the ratio of non-zero blocks).

τ, λ	comp. ratio	decomp. time(sec)(I,P)
$\tau = 10\%, \lambda = 10\%$	35 : 1	8.31 , 0.84
$\tau = 10\%, \lambda = 1\%$	96 : 1	8.18 , 0.13
$\tau = 5\%, \lambda = 10\%$	58 : 1	7.89 , 0.81
$\tau = 5\%, \lambda = 1\%$	188 : 1	7.67 , 0.12

Table 3: *test results on ocean temperature change data*

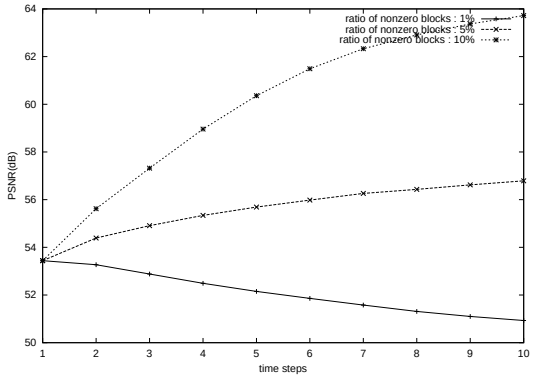

Figure 4: *PSNR of ocean data 10 reconstructed volumes in a same temporal group for* $\tau = 5\%$ *and* $\lambda = 1\%, 5\%, 10\%$.

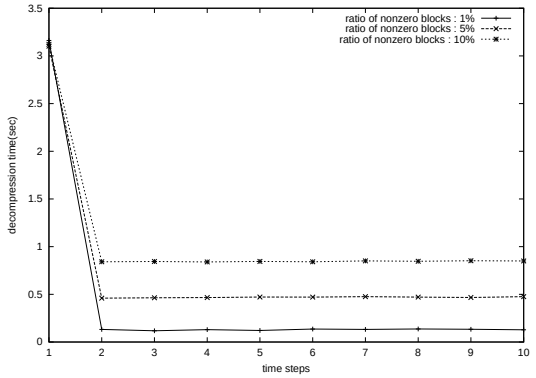

Figure 5: *decompression time of ocean data 10 volumes in a same temporal group for* $\tau = 5\%$ *and* $\lambda = 1\%, 5\%, 10\%$.

As you can see in the tables 2 and 3, compression ratio is highly dependent on τ(non-zero coefficients ratio) and λ(non-zero blocks ratio). While the decompression time of the intra-coded frame is quite big and is stable with regard to τ and λ, the decompression time of predictive frames is much smaller and depenent on λ. This result is well matched with our goal which is to reduce the number of blocks to update and minimize run-time decompression time. Figure 6 shows the quality of images over three separated time steps with different image quality. Although we can recognize some severe artifacts in $\lambda = 1\%$, and minor artifacts in $\lambda = 10\%$, we can achieve significant storage save and real-time playback of time-varying volume data by using our compression and real-time decompression scheme.

Figure 4 shows the PSNR of each frame of reconstructed volume with different λ value. Since frame 1 is intra-coded frame, no block truncation is performed which is why compression with different λ has same PSNR in intra-coded frame. The graph implies that in ocean data set, 5% and 10% of non-zero blocks are more than enough to keep similiar

image quality as intra-coded frame because PSNR increases over time even if we only update 5% or 10% of significant blocks.

Figure 5 provides the decompression time of each frame in the same temporal group with regard to different λ. Since most of blocks in the intra-coded frame need to be decompressed, reconstruction of intra-coded frame is slow. On the other hand, the reconstruction of predictive frames takes much less because the number of updated blocks are small.

Figure 8 shows one frame of ocean temperature with two different transfer functions. While the second image shows the flow of warm water, the first image visualizes water of a wide range of temperatures. Both images are rendered from the same frame compressed with $\tau = 5\%$ and $\lambda = 5\%$ and little artifact can be recognized.

6. Conclusion

Since large time-varying volume data has lots of coherence, compression is necessary for both saving storage space and improving the performance of visualizing the time-varying data. However, standard video compression technique using encoding every blocks can result in poor run-time decompression speed. From this motivation, we achieved our goals : (i) high compression (ii) minimal image distortion, and (iii) real-time decompression, by truncating insignificant blocks and wavelet coefficients. We also showed that updating only small number of significant blocks can provide much faster decompression while keeping a high compression ratio and maintaining image quality.

Acknowledgements

This research is supported in part by NSF ACI-9982297, CCR-9988357, DOE-LLNL/SANDIA BD4485-MOID-1, NSF-NPACI-UCSD 10181410 and Texas Higher Education Coordination Board BD 781.

References

1. E. B. Lum, K.-L. Ma, and J. Clyne. Texture Hardware Assisted Rendering of Time-Varying Volume Data. In *IEEE Visualization 2001 Conference* . 2001. 2
2. D.L. Gall. MPEG: A Video Compression Standard for Multimedia Applications. In *Communications of the ACM*, 34, 4, April 1991. 2
3. W. B. Pennebaker and J. L. Mitchel. JPEG Still Image Data Compression Standard. *Van Nostrand Rheinhold*, New York, 1993. 2
4. S. Guthe, and W. Strasser. Real-Time Decompression And Visualization Of Animated Volume Data. In *IEEE Visualization 2001 Conference* . 2001. 2
5. Sengupta, M. Hilton, and B. Jawerth. A computationally fast wavelet-based video coding scheme. In *Digital Video Compression on Personal Computers: Algorithms and Technologies* , volume 2187 of *Proceedings of the SPIE*, pages 152-157. SPIE, Bellingham, WA, 1994. 2
6. H.-W. Shen, L.-J. Chiang, and K.-L. Ma. A Fast Volume Rendering Algorithm for Time-Varying Fields Using a Time-Space Partitioning(TSP) Tree. In *IEEE Visualization 1999 Conference*, pages 371-378, 1999. 2
7. K.-L. Ma, and D. Camp. High Performance Visualization of Time-Varying Volume Data over a Wide-Area Network. In *Supercomputing 2000 Conference*, November 4-10, 2000. 2
8. K.-L. Ma, D. Smith, M.-Y. Shih, and H.-W. Shen. Efficient Encoding and Rendering of Time-Varying Volume Data. ICASE Report No. 98-22, 1998. 2
9. D. Ellsworth, L.-J. Chiang, and H.-W. Shen. Accelerating Time-Varying Hardware Volume Rendering using TSP Trees and Color-Based Error Metrics. In *Proceedings of Volume Visualization and Grphics Symposium 2000*, pages 119-128, 2000. 2
10. C. Bajaj, I.Ihm, and S. Park. 3D RGB Image Compression for Interactive Applications. *ACM Transactions on Graphics*, Vol. 20, No. 1, pages 10-38, January 2001. 2

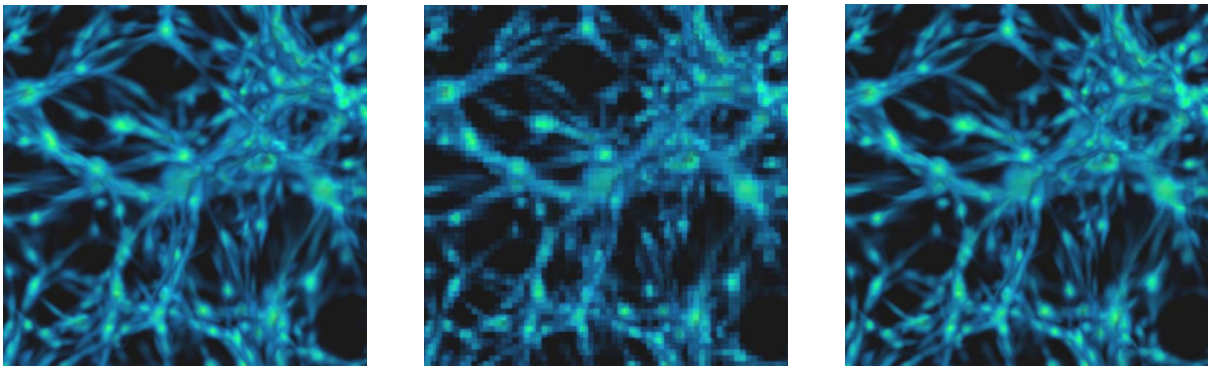

Figure 6: *Universe gas density data. Original volume, compressed volume with* $\tau = 1\%, \lambda = 5\%, comp.ratio is 145:1$, *and compressed volume with* $\tau = 5\%, \lambda = 5\%$, *comp. ratio is 96:1 in order.*

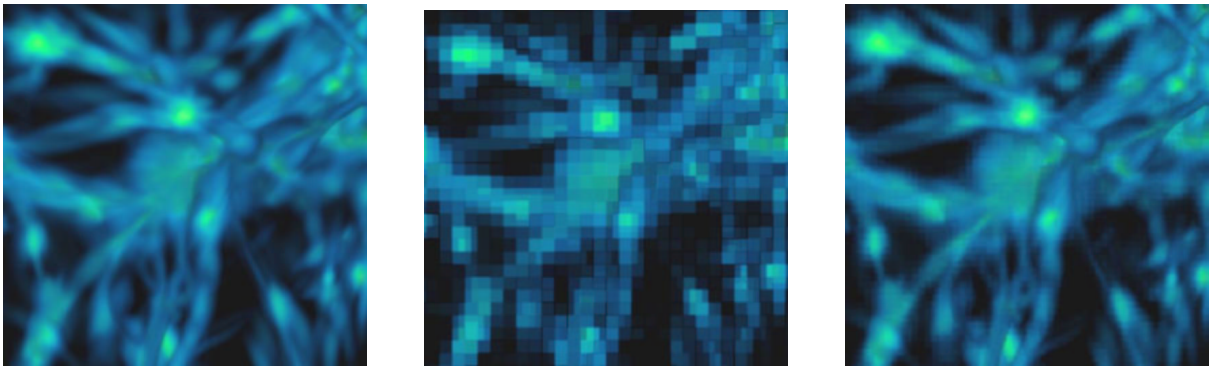

Figure 7: *Universe gas density data. Zoomed image from figure 6.*

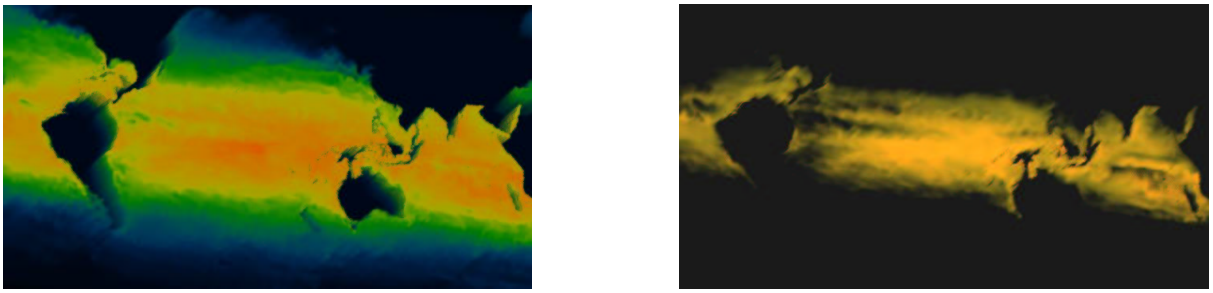

Figure 8: *A snapshop of ocean temparature change displaying with two different transfer function.* $\tau = 5\%$ *and* $\lambda = 5\%$, *compression ratio is 91.6:1*