Time-varying Volume Compression in Spatio-temporal Domain

Kun Zhao 1, Naohisa Sakamoto 2, and Koji Koyama 2

1 Graduate School of Engineering, Kyoto University, Japan
2 Institute for the Promotion of Excellence in Higher Education, Kyoto University, Japan

Abstract

Data compression is always needed in large-scale time-varying volume visualization. In some recent application cases, the compression method is also required to provide a low-cost decompression process. In this paper, we propose a compression scheme for large-scale time-varying volume data using the spatio-temporal features. With this compression scheme, we are able to provide a proper compression ratio to satisfy many system environments (even a low-spec environment) by setting proper compression parameters. After the compression, we can also provide a low-cost and fast decompression process for the compressed data. Furthermore, we implement a specialized particle-based volume rendering (PBVR) [2] to achieve an accelerated rendering process for the decompressed data. As a result, we confirm the effectiveness of our compression scheme by applying it to the large-scale time-varying turbulent combustion data.

Keywords - compression, time-varying, volume data, visualization

1 Introduction

Since the time-varying volume data from many scientific simulations always have a very large data size, a good method to compress the volume data is always needed to achieve an efficient visualization. In some recent application cases, the compression method is also required to provide a low-cost decompression process.

To solve this problem, we propose a compression scheme for large-scale time-varying volume data in spatio-temporal domain. In our proposed compression scheme, there are two compression processes performed to separately utilize the spatial features and temporal features of the time-varying volume data. With this compression scheme, we are able to compress time-varying volume into a proper compression ratio to satisfy many system environments by setting proper compression parameters. After the compression, we can also provide a low-cost and fast decompression process for the compressed data. Furthermore, we implement a specialized particle-based volume rendering (PBVR) [2] to achieve an accelerated rendering process for the decompressed data.

2 Related Work

In the previous researches, there are many volume compression methods proposed [3, 4, 5, 6] to achieve the time-varying volume compression. However, these compression approaches always need a costly decompression process or would generate a large decompression size, which would not fit for the application requiring low-cost decompression.

3 Proposed Method

Our proposed compression scheme contains two compression processes: spatial domain compression and temporal domain compression, which to utilize the spatial features and temporal features.

3.1 Spatial Domain Compression

The first process is the spatial domain compression. This method has been proposed in our previous work [1]. With the structured volume data, we use the two-level division to reduce the vertex number firstly, and fast cubic b-spline to reconstruct the divided volume (Figure 1). The two-level division first divides the volume into many blocks with the same cell number on each side (block size), and then subdivides every block into 24 tetrahedra. After that, the value of newly generated vertex (black dots in Figure 1) is evaluated by fast cubic b-spline. We hereinafter call this structure as “24-tetrahedra” mesh.

![Image](304x404 to 358x412)

**Fig. 1. Spatial domain compression with block size = 3.**

3.2 Temporal Domain Compression

In many cases, time-varying volume data always keep an invariant mesh between consecutive time steps. Obviously, such a kind of data would still keep an invariant mesh after the spatial domain compression. Since invariant-mesh data often contains much temporal coherence, some parts of the volume may have a strong similarity for the consecutive time steps. Hence, we can achieve a second volume compression by deleting these similarity parts based on the compression result of the spatial domain compression.

In our method we first calculate the coherency property for each vertex into a vertex table (VT) and then delete these coherency vertices to compress the volume. Assume two consecutive time steps as the previous step and current step; we calculate the value of vertex table for the ith vertex as:

\[ VT_i = \begin{cases} 1, & |V_i^{prev} - V_i^{curr}| < \text{tolerance} \\ 0, & \text{else} \end{cases} \]

(1)

Where \( V_i^{prev} \) represents the ith vertex value of the previous time step, and \( V_i^{curr} \) represents the ith vertex value of the current time step. Obviously, “1” means the vertex is coherency, and “0” means the vertex is not coherency. We allow user to set a tolerance rate \( \varepsilon \), and our system will calculate the tolerance as:

\[ \text{tolerance} = \varepsilon \times (V_{max}^{prev} - V_{min}^{prev}) \]

(2)

Here, \( V_{max}^{prev} \) and \( V_{min}^{prev} \) respectively means the maximum value and minimum value of the previous step.
3.3 Decompression and Visualization

With compressed data, the decompression process only needs to copy the coherency vertices from the previous time step. Note that our decompression is not to reconstruct the compressed data into the original mesh but to reconstruct it into the “24-tetrahedra” mesh. That is because the original mesh has a large data size, which is very hard to handle. If we decompress the compressed data into the original size, the compression would be meaningless.

After the decompression process, we use PBVR to render the decompressed “24-tetrahedra” mesh. PBVR [5] is a stochastic rendering method, which first generates particles for the volume data and then projects particles to get the rendering image. However, the particle generation would cost much time for the large-scale volume data. In fact, since the time-varying volume data often contains much temporal coherence, the coherence is also able to accelerate the particle generation of PBVR. Since the temporal coherence has been calculated into vertex table, we generate particles by referring VT. With the temporal coherence, we skip the coherency part and only generate particles for the none-coherency part. This can accelerate particle generation time obviously.

4 Experiments and Results

In this paper, we apply our method to the large-scale time-varying dataset obtained from a turbulent combustion simulation. There are 480×720×120 voxels composed in float type, and a total of 122 time steps (18.9GB). The experiment is conducted with an Intel Core i7-2820QM CPU (2.3 GHz), an NVIDIA GeForce GTX580M 2GB GPU, and 16GB system memory. The operation system is Ubuntu 12.04 LTS.

In our experiment, the compression ratio has been calculated as original data size over compressed data size. The compression ratios with different block size (BS) and tolerance rates (TR) are shown in Table 1. With the compressed data, the decompression time is shown in Table 2.

| TR  | BS | 2    | 3    | 4    | 5    |
|-----|----|------|------|------|------|
| 10^7|     | 3.95:1 | 13.40:1 | 31.65:1 | 61.35:1 |
| 10^6|     | 4.14:1 | 14.06:1 | 33.11:1 | 64.52:1 |
| 10^5|     | 4.39:1 | 14.88:1 | 35.09:1 | 68.03:1 |
| 10^4|     | 4.92:1 | 16.67:1 | 39.37:1 | 76.34:1 |

Table 1. Compression ratio

| TR  | BS | 2    | 3    | 4    | 5    |
|-----|----|------|------|------|------|
| 10^7|     | 6.63s | 6.07s | 5.92s | 5.86s |
| 10^6|     | 6.78s | 6.17s | 6.00s | 5.96s |
| 10^5|     | 6.88s | 6.16s | 6.01s | 5.97s |
| 10^4|     | 7.02s | 6.37s | 6.24s | 6.18s |

Table 2. Decompression time

After the decompression for the volume data, we render the decompressed data with our specialized PBVR process. Table 3 shows the particle generation time with the decompressed volume data. Here, the block size is 2. Since different tolerance rate would also generate different temporal coherence for the compressed volume data, we show the particle generation time for different tolerance rate. Here, “N/A” means the particle generation time without using temporal coherence.

| TR  | N/A | 10^7 | 10^6 | 10^5 | 10^4 |
|-----|-----|------|------|------|------|
| Particle Generation | 3.947.9s | 3.256.8s | 3.214.5s | 3.149.1s | 3.149.1s |

Table 3. Particle generation time

5 Discussion and Conclusion

From the experimental results we could see, our compression scheme could compress the time-varying data into different size with different compression parameters - block size and tolerance rate, which are used to control the calculation precise for spatial features and temporal features. Thus, we could provide a proper compression ratio to satisfy many system environments (even a low-spec environment) by setting proper compression parameters. This is a main feature of our compression scheme. Moreover, another main feature is that a fast decompression speed could be provided for the spatio-temporally compressed data. This could be observed in Table 2, where the decompression process cost only about 6 seconds or 7 seconds for all 122 time steps of the time-varying combustion data. After decompression, our specialized PBVR to utilize the temporal coherence also shows efficiency that the particle generation time could be accelerated remarkably (see Table 3). This accelerated rendering process is another feature of our compression scheme.

In this paper, we have proposed a volume compression scheme for the large-scale time-varying volume data using spatio-temporal features. With this compression scheme, we could provide a proper compression ratio to satisfy many system environments (even a low-spec environment) by setting proper compression parameters. Moreover, since our compression scheme does not need special process such as inverse transfer during the decompression, we could provide a low-cost and fast decompression. Furthermore, we also implemented a special method into PBVR to accelerate the particle generation process when we render the spatio-temporally compressed volume. These features are verified in an experiment for the time-varying combustion data.

References

[1] Kun Zhao, Naohisa Sakamoto, Koji Koyama: A Volume Compression Scheme Based on Block Division with Fast Cubic B-spline Evaluation, In the Proceedings of the Asia Simulation Conference 2012(AsiaSim2012), 2012
[2] T. Kawamura, N. Sakamoto and K. Koyama: A Level-of-Detail Rendering of a Large-Scale Irregular Volume Dataset Using Particles, Journal of Computer Science and Technology, Vol.25, No.5, pp. 905-915, 2010
[3] Yun Jung, Ebert, D.S., Gaither, K.: Time-Varying Data Visualization Using Functional Representations, Visualization and Computer Graphics, IEEE Transactions on, Vol.18, No.3, pp.421-433, 2012
[4] Schnerder J., Westermann R.,Compression domain volume rendering. In Proceedings of IEEE Visualization, pp. 293-300, 2003
[5] Jorg Mensmann, Timo Ropinski, and Klaus Hinrichs: A GPU-Supported Loss-less Compression Scheme for Rendering Time-Varying Volume Data, Volume Graphics Eurographics Association, pp. 109-116, 2010
[6] M. Weiler, R. P. Botchen, S. Stegmeier, T. Ertl, J. Huang, Y. Jang, D. S. Ebert, and K. P. Gaither. Hardware-assisted feature analysis of procedurally encoded multifield volumetric data. Computer Graphics and Applications, 25(5), pp. 72–81, 2005