Block Division with Fast Cubic B-spline Reconstruction

Kun Zhao [1], Naohisa Sakamoto [2], and Koji Koyamada [2]

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

1 Introduction

In the recent years, the high performance computing (HPC) environment has been improved. Thus, the volume dataset generated from HPC often resulted in the high resolution with respect to the spatial and temporal dimension. This situation requires a good visualization method, which can handle such a large-scale volume dataset.

In this paper, we propose a volume reconstruction method that employs tetrahedral cells generated at each sub-volume. The sub-volume is defined by applying a blocking operation to an original volume with a block size. Additional vertices in the tetrahedral cells are calculated by using a cubic b-spline interpolation function calculated in an original volume. We confirmed the effectiveness by calculating a compression ratio for time-varying volume dataset.

2 Related Work

Since the volume data is defined at discrete points, which are usually sampled from a continuous scalar field, many methods have been proposed for seeking a continuous and accurate volume representation. Among them, the successful application of b-spline in signal and image processing field [1] was promising and an efficient algorithm to calculate the b-spline coefficients has been proposed [2][3]. Bajaj C.L.[4] et al have verified that the cubic b-spline is also a perfect fit for volume reconstruction since it would yield less aliasing of high frequency compared to a quadratic b-spline and also keep a second-order continuity.

3 Volume Reconstruction System

In our proposed reconstruction system, we first divide the volume into many blocks with the same block size, and then subdivide every block into 24 tetrahedra. Afterwards, the value of the vertices newly generated would be evaluated by fast cubic b-spline algorithm.

3.1 Block Division

With the original volume data, the block division is done with two levels.

The first-level division is to divide it into many blocks with the same block size (left figure of Figure 1). Obviously, this would decrease the number of the vertices greatly but it can also make a great data loss. As a result, we also propose a second-level division to subdivide the block into 24 tetrahedra (right figure of Figure 1) so that the area inside the block can also be also represented well. Moreover, since the block is subdivided into tetrahedra, it would not lead too much value gap.

With the block division, center points of every faces and gravity point of the block (black dots in Figure 1) would be needed. As a result, an evaluation method is needed to evaluate these values.

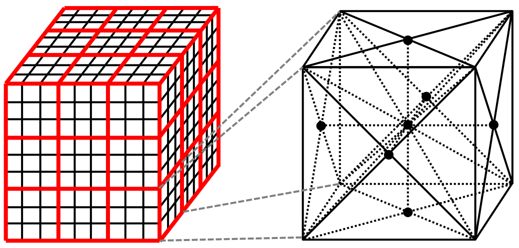


Fig.1. Block division with block size = 3

3.2 Fast Cubic B-spline Reconstruction

Ideally, we think the value of every newly generated point should keep continuity with the neighbour vertex points so that less data gap would be generated. Since the good performance of cubic b-spine, we utilize it to evaluate the value of the newly generated vertices. The basic formula of the cubic b-spine method can be written as:

$$s(x) = \sum_{i=1}^{n-1} c_i \beta^3(x-i) \tag{1}$$

where the $s(x)$ means the reconstructed signal, c_i means the b-spline coefficients and $\beta^3(x)$ means the b-spline kernel. Here, the b-spline kernel can be defined as:

$$\beta^3(x) = \begin{cases} \frac{2}{3} - x^2 + \frac{1}{2}|x|^3, & 0 \le |x| < 1 \\ \frac{1}{6}(2-|x|)^3, & 1 \le |x| < 2 \\ 0, & 2 \le |x| \end{cases} \tag{2}$$

And for the coefficients of $c_1, c_2, \ldots, c_{n-1}$, we use the fast algorithm proposed by Bajaj C. L. et al [4]. The computation of coefficients is done with the recursive process with the initial values of (4). Here, $\{f(k)\}$ is the original scalar values, and superscript of + and – are used to specify the recursive process.

$$c_0^+ = -\frac{1}{1-z_1^{2n}} \sum_{k=1}^{n-1} f(k) z_1^k \left(1 - z_1^{2(n-k)}\right), c_n^- = 0 \tag{4}$$

And then, the recursive process

$$c_k^+ = f(k) + z_1 c_{k-1}^+, \quad k = 1,2,\ldots,n-1$$
$$c_k^- = z_1(c_{k+1}^- - c_k^+), \quad k = n-1,\ldots,2,1$$

yields exact B-spline coefficients $c_i = 6c_i^-$, where $z_1 = \sqrt{3} - 2$.

The computation of c_0^+ can be accelerated by neglecting small terms. Given an error control tolerance ϵ, we compute

$$K = \left\lceil \frac{\log(\epsilon)}{\log(|z_1|)} \right\rceil$$

And if $n > K$, c_0^+ would be replaced with

$$c_0^+ = -\frac{1}{1-z_1^{2n}} \sum_{k=1}^{n-1} f(k) z_1^k \tag{5}$$

which can be computed using the Horner scheme for evaluating a polynomial. This could accelerate the computation remarkably.

4 Experiments and Results

In this paper, we apply our method on the time-varying dataset obtained from a turbulent combustion simulation. The grid structure is Cartesian with uniform spacing. There are 480×720×120 voxels, and a total of 122 time steps. The value of the vertices is composed in the float type. The experiment results with different block size are shown as the Table 1. And the rendering results for different block size by Particle-Based Volume Rendering (PBVR) [5] with phong shading are shown in Figure 2 (as the limitation of space, the image of block size equals 4 is omitted.). PBVR is a stochastic rendering method, which can provide a good performance for both structured volume data and unstructured volume data by generating particles from the volume data. The data used here is the Y modality of time step 35. The experiment is conducted with Intel Core i7-2820QM CPU (4 cores), NVidia GeForce GTX580M 2GB GPU, and 16GB RAM.

Here, we calculate the compression ratio by calculating compressed data size over original data size. From the Table 1 we can see, even though tetrahedral cells are increased, the compression ratio for all time steps performs very well since the added coordinate and connection information for the tetrahedral cells are needed only once for all steps. Moreover, the average error is also computed from:

$$E = \frac{\sum_{i=1}^{N} \left| V_{ori}^{i} - V_{com}^{i} \right|}{N}$$

where, N is the vertices number of the original volume, V_{ori}^{i} is the normalized value of the original volume at vertex i, and V_{com}^{i} is the normalized vertex value also at vertex i interpolated linearly from the compressed volume.

5 Discussion and Conclusions

The result of Figure 2 shows that when the block size is smaller than 5 our reconstruction system would provide a clear and smooth image quality. Moreover, as we use the cubic b-spline as the evaluation method to evaluate the newly generated vertices after the block division, the comparison with using linear interpolation as the evaluation method is also shown in Figure 3 (local area of the same data to Figure 2 with block size 2). We can see that the cubic b-spline evaluation can actually provide a good image quality.

On the other hand, from equation (1) we can see, the cubic b-spline coefficients would have a $(n-1)$ data size, where n is the number of vertices of the original volume. As a result, after the reconstruction is done, the coefficients of the cubic b-spline are deleted to save the storage space. However, if we also take these coefficients into the visualization process of the reconstructed volume, the image quality might be even better because everywhere inside the tetrahedra cell can be interpolated by cubic b-spline. This approach would be optimal if we can also compress the coefficients so that it would not cost so much storage space. This would be done in our future work.

In this paper, we proposed a volume reconstruction method with block division based on the fast cubic b-spline evaluation. From the above experiments and results, we can see our volume reconstruction system can provide a good compression ratio especially for time-varying dataset with a good image quality and a less compression error.

References

[1] H.S. Hou and H.C. Andrews. Cubic splines for image interpolationand digital filtering. IEEE Trans. Acoust., Speech, and Signal Processing, 26, 852–864, 1978.

[2] M. Unser, A. Aldroubi, and M. Eden. B-spline signal processing: Part I-theory. IEEE Trans. Signal Processing, 41:821–832, 1993.

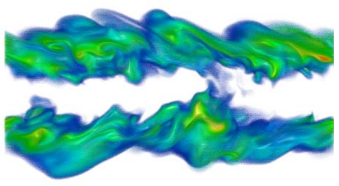
[3] M. Unser, A. Aldroubi, and M. Eden. B-spline signal processing: Part II-efficient design and applications. IEEE Trans. Signal Processing, 41, 834–848, 1993.

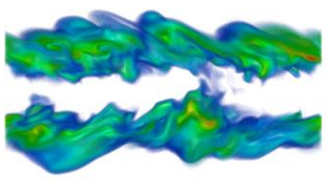
[4] C. Bajaj, G. Xu, Q.Zhang, Bio-Molecule Surfaces Construction Via a Higher-Order Level Set Method. Journal of Computer Science and Technology. 23(6), 1026-1036, 2008.

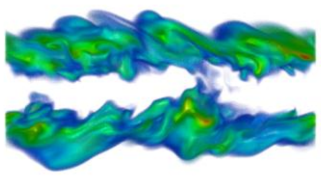
[5] T. Kawamura, N. Sakamoto and K. Koyamada, A Level-of-Detail Rendering of a Large-Scale Irregular Volume Dataset Using Particles, Journal of Computer Science and Technology, Vol.25, No.5, 905-915, 2010.

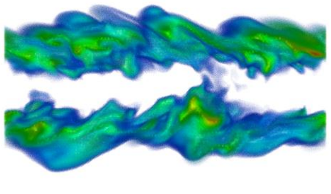
Table 1. Reconstruction with different block size ($\epsilon = 1.0 \times 10^{-9}$)

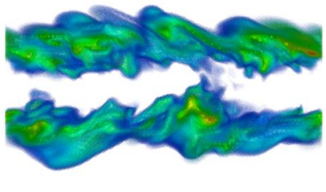
Block Size	Original Volume	2	3	4	5	6	7
Number of Vertices	41,472,000	26,165,461	7,789,241	3,301,531	1,698,313	987,421	624,347
Number of Tetrahedras	0	124,416,000	36,864,000	15,552,000	7,962,524	4,608,000	1,899,656
Compression Ratio for All Steps	100%	74.5%	22.1%	9.4%	4.8%	2.8%	1.7%
Processing Time	N/A	20.886s	11.581s	5.174s	4.609s	3.409s	3.389s
Average Error (E)	N/A	0.044%	0.081%	0.11%	0.18%	0.22%	0.30%


(a) Original data


(b) Block size =2


(c) Block size = 3


(d) Block size = 5


(e) Block size = 6

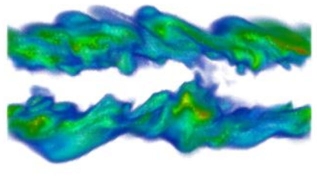
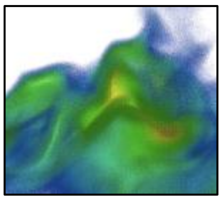

(f) Block size = 7

Fig.2 Rendering Result with PBVR for different block size


(a) Cubic b-spline evaluation

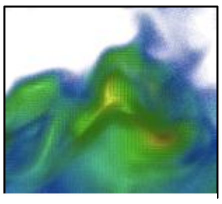

(b) Linear evaluation

Fig.3 Comparison with linear evaluation (block size = 2)