Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Computers & Graphics 28 (2004) 279–288

COMPUTERS & GRAPHICS

www.elsevier.com/locate/cag

Technical section

Time-critical rendering for time-varying volume data

Shih-Kuan Liao[a,b,*], Jim Z.C. La[a], Yeh-Ching Chung[a]

[a] Department of Information Engineering, Feng-Chia University, Taichung, Taiwan 407, ROC
[b] Department of Information Management, National Taichung Institute of Technology, Taichung, Taiwan 404, ROC

Abstract

In this paper, we present a novel method to meet the time-critical requirement in rendering time-varying volume data. In time-critical rendering, the rendering is demanded to be completed in a given time budget. Our approach is modified from the differential volume rendering, which updates only the changed pixels instead of all pixels on the image plane. The level of modification (LOM) is used to measure the degree of modification of the changed data between consecutive time steps. To meet the time-critical requirement, the presented method chooses the most important changed pixels to be updated. The experimental results show that our method is very good in both controlling the rendering time to meet the time constraint and preserving the important features of a data set within a limited time budget.

Keywords: Animation; Display algorithm; Volumetric; Ray casting; Time-varying volume data; Time-critical rendering

1. Introduction

Volume rendering includes a variety of widely used techniques to explore volume data in a variety of applications [1–3]. With this technique, a 3D volume data set is projected onto a 2D image plane to reveal the internal structure of the data set. Due to the large size of data, rendering volume data is usually a time-consuming task. In spite of the heavy burden, the rendering time for an image frame is sometimes constrained within a time budget, for example, in interactive or real-time rendering. The time-critical rendering usually has to trade image quality for rendering speed.

A time-varying volume data (TVVD) consists of a sequence of volume data sets in consecutive time steps. Rendering TVVD yields an animation presenting the evolution of the data set under investigation. When TVVD is rendered in batch, generating an image of good quality is the most concerned issue. However, when TVVD is rendered interactively, a constant frame rate, or a constrained rendering time for each time step, is more important.

Although a lot of works on time-critical rendering and TVVD have been presented previously, the research on time-critical TVVD rendering is scarce. In this paper, we focus on the time-critical rendering for TVVD. The goal is to maximize the image quality under the constraint of a given time budget. Based on the existing differential volume rendering method (DVRM) for TVVD, we develop a time-critical DVRM (TCDVRM).

Several TVVD rendering methods use the data coherency of TVVD to improve the rendering performance. For such approaches, coherent data inside TVVD are retained and re-used. On the other hand, only the dynamic data, or the changed fraction of data, are processed. Shen et al [4] proposed the DVRM, which determines the changed pixels due to changed voxels and updates only the changed pixels in consecutive time steps. Anagnostou et al. [5] applied the similar concept with the shear warp factorization in rendering 4D volume data. In Ref. [6], an enhanced DVRM, the two-level DVRM (TLDVRM), was presented to accelerate the determination of changed pixels with the aid of the second-order difference. Hierarchical data structure such as octree is also widely used to manipulate the

*Corresponding author. Department of Information Engineering, Feng-Chia University, Taichung 407, Taiwan. Tel.: +886-4-24517250; fax: +886-4-24516101.
E-mail addresses: skliao@ntit.edu.tw (S.-K. Liao), lai@fcu.edu.tw (J.Z.C. La), ychung@fcu.edu.tw (Y.-C. Chung).

0097-8493/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2003.12.012

coherency in TVVD [7,8]. In Ref. [9], a type of hierarchical data structure, the time-space partitioning (TSP) tree, was used to utilize both the spatial and the temporal coherency effectively. This method allows flexible degrading of image quality for faster rendering speed. However, the reduction in rendering time is not predictable. Therefore, the method cannot control the rendering time to meet the constraint of time budget. Other approaches for TVVD rendering include parallel processing [10], hardware assistance [1], and transformation [11].

The level-of-detail (LOD) selection is a popular approach to render polygon meshes [12,13]. The LOD selection is also applied to render volume data [14,15]. Trading image quality to gain rendering speed is possible with a variety of LOD selection algorithms. In such approaches, the original data set is transformed into a set of data sets of different resolutions, where a coarser resolution loses more details and has the lower LOD. When the rendering is going on, an appropriate resolution, or an LOD, is selected according to some specific criteria. In spite of their varieties, these works share a common concept in degrading rendering quality to accelerate rendering speed. Other approaches on time-critical volume rendering include inter-frame feedback [16] and simplification on both image and object spaces [17].

Although the information of LOD is useful for static volume data, it is not suitable for coherence-based methods. The reason is that the LOD is applied to the volume data itself rather than the changed fraction, which should be handled by a coherence-based method. Therefore, we use the level of modification (LOM) to represent the degree of modification of changed data instead of using the LOD to indicate the resolution of volume data. Changed data with the higher LOM are more important than those with the lower LOM.

The developed time-critical rendering method for TVVD is modified from DVRM by adopting a control scheme that uses the LOM of changed data. Experimental results show that our method can effectively control the rendering time to meet the time-critical requirement and produce good animations that reveal the important dynamic features of data.

The rest of this paper is organized as follows. In Section 2, the background of DVRM is reviewed. In Section 3, the presented method is described in detail. In Section 4, the experimental results of the presented method are given. Finally, conclusions are presented in Section 5.

2. Background

Ray casting is widely used in volume rendering to generate an image of good quality. In the method, a ray is fired for each pixel to penetrate the volume data. At constant sampling interval, sampling points along the ray are given their interpolated gradients and sample values from the neighboring voxels. Given transfer functions, the sample values and gradients at sampling points are mapped to color and opacity values. Finally, the color and opacity of sampling points along the ray are accumulated to the pixel color [18]. DVRM renders TVVD based on the ray casting method. DVRM selects the changed pixels whose values are changed in consecutive time steps. Since each ray is processed independently from one another, only the changed pixels are updated while the rest are left unchanged. DVRM consists of two phases: the static phase and the dynamic phase. In the static phase, the changed voxels between consecutive time steps are extracted. The positions and values of changed voxels are stored in differential files. In the dynamic phase, the data are rendered. During the dynamic phase, the rendering parameters, such as view direction and transfer functions, are fixed. The first image is rendered in a regular method. For each of the following time steps, the differential files are loaded to update the volume data. The positions of changed pixels are then determined. At the end of the time step, changed pixels are updated by firing new rays. If only parts of pixels are changed, the time of ray casting is reduced. However, the reduction of ray casting may be compensated with the overhead of the determination of changed pixels. When the number of changed voxels is large, DVRM may not be efficient.

In a previous work [6], we presented the TLDVRM to accelerate the determination of changed pixels. Some of the concepts used in developing TLDVRM are useful for the time-critical TVVD rendering. TLDVRM filters out the overlapped changed voxels and extracts the difference of changed voxel positions between consecutive differential files. The extracted difference information is referred to the second-order difference (SOD). Using SOD, the redundant computation of determining changed pixels due to overlapped changed voxels can be omitted. As a result, TLDVRM can determine the changed pixel positions more efficiently. For the convenience of presenting our approach, Definitions 1 and 2 are given below. The notations used in this section are listed as follows.

V_t	the volume data in time step t
(x, y, z)	a voxel
(x, y, z, d_t)	a changed voxel (x, y, z) and its voxel value d_t in time step t
DF_t	the differential file consisting of (x, y, z, d_t), changed voxels and their voxel values, in time step t
$P(DF_t)$	the set of changed voxels in DF_t
(r, s)	a pixel
B	the size of a disk I/O sector

C_f the time to read a sector
C_g the time to compute the gradient of a voxel
C_p the time spent for a changed voxel in the determination of changed pixels
C_c the time to process a sampling point
CS_t the sum of the numbers of sampling points of all of the changed pixels in time step t

Definition 1. A voxel (x, y, z) is said to influence a pixel (r, s) if the pixel value at (r, s) needs to be updated due to the changed voxel value at (x, y, z). We denote $J(x, y, z)$ as the set of pixels influenced by the voxel (x, y, z).

$J(x, y, z)$ depends on parameters such as the view direction and sampling method. For example, when discrete rays and zero-order interpolation are used, a changed voxel is projected onto the image plane and $J(x, y, z)$ includes the four pixels surrounding the projected point. When continuous rays and trilinear interpolation are used, the interpolation space that encompasses a changed voxel is projected onto the image plane and $J(x, y, z)$ includes the pixels located inside the projected region.

Definition 2. The number of changed voxels that influence pixel (r, s) in time step t is defined as $I_t(r,s) = |W|$, where $W = \{(x, y, z) | (r, s) \in J(x, y, z)$ and $(x, y, z) \in P(DF_t)\}$. We called $I_t(r, s)$ the *influence number* of pixel (r, s) in time step t.

If the influence number of each pixel is known, a pixel whose influence number is greater than zero is classified as a changed pixel. Influence numbers either can be calculated directly by checking all the changed voxels or can be determined by checking SOD. Fig. 1(a) illustrates that how a changed pixel's influence number is determined and Fig. 1(b) explains the process of determining changed pixels with influence numbers. Influence numbers imply that how seriously pixels are changed. Actually, a lot of changed pixels have very small influence numbers. That is, most of the voxels along the ray paths of these changed pixels are not changed, and only a few of them are changed.

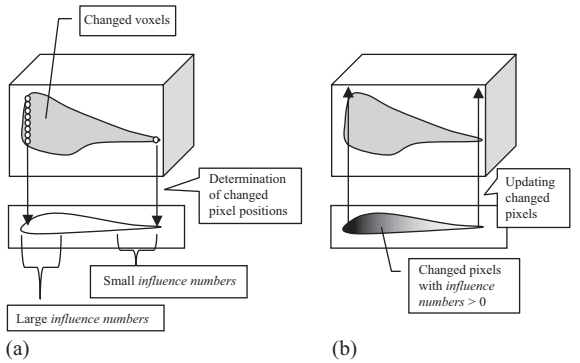

Fig. 1. (a) Changed voxels are projected onto the image plane to compute influence numbers of pixels. (b) A pixel with influence number >0 is classified as a changed pixel.

Brief discussion of DVRM is given below because the presented time-critical rendering method for TVVD is based primarily on DVRM. Each rendering process for a time step in DVRM is a pipeline, which composes of several stages: the stages to load files to update volume data, to compute the gradients, to determine the changed pixels, and to do ray casting. The rendering time of DVRM in time step t is therefore the sum of the processing time of all stages and is expressed by the following equation:

$$T_t(DVRM) = \lceil 4|P(DF_t)|/B \rceil \times C_f + |V_t| \times C_g + |P(DF_t) \times C_p + CS_t \times C_c, \quad (1)$$

where DF_t is the differential file in time step t; $|P(DF_t)|$ is the number of changed voxels in DF_t; B is the size of a disk I/O sector; C_f is the time to read a sector; $|V_t|$ is the number of voxels in time step t; C_g is the time to compute the gradient of a voxel; C_p is the time spent for a changed voxel in the determination of changed pixels; CS_t is the sum of the numbers of sampling points of all of the changed pixels; and C_c is the time to process a sampling point.

Eq. (1) predicts the rendering time of DVRM if all the parameters in the equation are known. Unfortunately, C_f, C_g, C_p, and C_c are machine and run time dependent and they are very difficult to be modeled before rendering. Fortunately, we are dealing with TVVD that spans perhaps hundred of time steps. We can measure actual C_f, C_g, C_p, and C_c in the first few time steps. For the rest of time steps, we can use these known C_f, C_g, C_p, and C_c. As a result, the rendering time of DVRM can be successfully predicted based on Eq. (1).

3. Time-critical DVRM

The presented method, TCDVRM, is modified from DVRM to fulfill the time-critical requirement. The overview of the proposed method is given below. The method consists of two phases:

(a) *The static phase*: This phase is the same as that of DVRM. Differential files are obtained in this phase.
(b) *The dynamic phase*: Rendering takes place in this phase. In each time step of this phase, the following jobs are performed sequentially:
  (1) *Initialization*: The necessary files are read; volume data are updated; and gradients are calculated.

(2) *Determination of a set of changed pixels*: The set includes as many "important" changed pixels as possible so that the estimated time to render these changed pixels will not exceed the time budget.
(3) *Ray casting for the changed pixels*: For each changed pixel in the set, a ray is cast.

The major distinction between our method and DVRM is that our method does not determine all changed pixels. The presented method gives up some of changed pixels to meet the restriction of time budget. The limited time should be spent for the most important changed pixels. In addition, the overhead of the determination of changed pixels should be reduced to the extreme. Several things are helpful in developing the method. First, the rendering time of DVRM can be estimated. This helps to determine the amount of rendering in advance. Second, the influence numbers of changed pixels are not uniform. This helps to give high priorities to select important changed pixels with large influence numbers. These concepts are integrated in the determination of changed pixels with large influence numbers. In the following, we will describe the algorithm of determining changed pixels with large influence numbers. Additional notations used in this section are listed below.

SV	the number of sampled changed voxels
$PS_t(r, s)$	the probability that (r, s) will be selected during the random sampling
$D(r, s)$	the number of sampled voxels that influence pixel (r, s) during the sampling process
$S(r, s)$	the number of sampling points along the ray of (r, s)
TS	the total number of sampling points to be cast
TB	the time budget for a time step

3.1. Level of modification

We use the LOM of changed data instead of the LOD of all volume data to control the rendering time. The LOD is a suitable index to control the rendering time of a single static volume data set. However, the most distinguishing characteristic of TVVD in contrast with a single static volume data set is the dynamic property. The difference between volume data in consecutive time steps is very important because it reveals the evolution of the data set. Therefore, we use the LOM to represent the degree of the change of a dataset or an image. The LOM may be in a variety of forms to express the dynamic property. For example, the amount of increment (or decrement) of a voxel value in a time step can be a kind of LOM. A changed voxel with a large amount of increment (or decrement) is changed more seriously than a changed voxel with a small amount of increment (or decrement). In other word, the former changed voxel is more important than the latter one.

In this paper, we use the influence number of a changed pixel as a metric of LOM to control the rendering time. It is noted that pixels with great variations between consecutive time steps are usually corresponding to pixels with larger influence numbers. That is, a changed pixel with a larger influence number is more important than a changed pixel with a smaller influence number. If the time budget does not allow all changed pixels to be updated, a changed pixel with the larger influence number will be selected in a higher priority. In a simplified example denoted as Example 1, assume that each changed voxel influences only one changed pixel. There are totally 161 changed voxels; 100 of them influence changed pixel $(r1, s1)$; 50 of them influence changed pixel $(r2, s2)$; 10 of them influence changed pixel $(r3, s3)$; and one of them influences $(r4, s4)$. The influence numbers $I_t(r1, s1)$, $I_t(r2, s2)$, $I_t(r3, s3)$, and $I_t(r4, s4)$ are 100, 50, 10, and 1, respectively. If the time budget allows only one changed pixel to be updated, $(r1, s1)$ is the best candidate. If the time budget allows two changed pixels to be updated, $(r2, s2)$ is the second candidate beside $(r1, s1)$. When the time budget allows all changed pixels to be updated, $(r3, s3)$ and $(r4, s4)$ are the third and last candidates, respectively.

3.2. Determination of changed pixels

We use a novel approach to determine a set of the most important changed pixels with larger influence numbers without calculating their influence numbers in advance. This method samples changed voxels randomly. The changed pixels that are influenced by the sampled voxels are recorded and the time of ray casting for these changed pixels is estimated. The method stops sampling and begins ray casting when the estimated time of ray casting reaches the time budget. Although influence numbers are not calculated at all, the random sampling process will inherently select a changed pixel with the larger influence number in a higher probability.

Assume that the number of sampled changed voxels is SV. For a changed pixel (r, s) with influence number $I_t(r, s)$ in time step t, the probability that (r, s) will be selected using random sampling is denoted as $PS_t(r, s)$. $PS_t(r, s)$ can be determined as follows.

(1) *When* $SV + I_t(r, s)$ *is not greater than* $|P(DF_t)|$: The number of combinations to select SV changed voxels from $|P(DF_t)|$ changed voxels is $C(|P(DF_t)|, SV)$. $C(|P(DF_t)| - I_t(r,s), SV)$ is the number of combinations to select SV changed voxels from $|P(DF_t)|$ changed voxels excluding the ones that influence pixel (r, s). The

probability $PS_t(r, s)$ is expressed as follows:

$$PS_t(r,s) = 1 - \frac{C(|P(DF_t)| - I_t(r,s), SV)}{C(|P(DF_t)|, SV)}, \quad SV + I_t(r,s) |P(DF_t)|, \tag{2}$$

where $C(|P(DF_t)|-I_t(r,s),SV)/C(|P(DF_t)|,SV)$ is the probability that (r, s) is not selected. From Eq. (2), we can conclude that a larger $I_t(r, s)$ will result in the larger $PS_t(r, s)$.

(2) *When $SV+I_t(r, s)$ is greater than $|P(DF_t)|$*: In this condition, it is impossible to avoid selecting at least one of the $I_t(r, s)$ changed voxels that influence (r, s). Therefore, $PS_t(r, s)$ equals 1.

In the simplified example discussed in the previous section, $|P(DF_t)|$ is 161. The probabilities that $(r1, s1)$, $(r2, s2)$, $(r3, s3)$, and $(r4, s4)$ will be selected in the first 10 sampling processes are listed in Table 1. From Table 1, we can find that the most important changed pixels will be selected with very high probability using very few sampled changed voxels. By bypassing the calculation of influence numbers and the consequent operations such as sorting and filtering, the method is quite efficient in determining the most important changed pixels with large influence numbers. As more and more changed voxels are sampled, the less important changed pixels will also be selected. Although not presented in Table 1, all changed pixels will be selected definitely when SV is finally large enough.

3.3. Rendering time estimation

We now describe the condition to stop sampling changed voxels so that the time-critical requirement can be satisfied. As more and more changed voxels are sampled, more and more changed pixels will be selected. In order to estimate the time required to update an image by ray casting, the number of sampling points being cast for these changed pixels must be known. Let SV be the number of sampled changed voxels, $D(r, s)$ be the number of sampled voxels that influence pixel (r, s) during the sampling process, $S(r, s)$ be the number of sampling points along the ray of (r, s), and TS be the total number of sampling points to be cast. As illustrated in Fig. 2, $S(r, s)$ is not uniform. TS is calculated during the sampling process by accumulating the $S(r, s)$ of (r, s) whose $D(r, s)$ equals 1 (when (r, s) is determined at the first time). At each moment when a changed voxel is sampled, SV and TS will increase. If the sampling is stopped and the ray casting is started, the rendering time of this time step, $T_t(TCDVRM)$, may be calculated using the following equations, which are modified from Eq. (1).

$$T_{elp} = \lceil 4|P(DF_t)|/B \rceil \times C_f + |V_t| \times C_g + SV \times C_p, \tag{3}$$

$$T_t(TCDVRM) = T_{elp} + TS \times C_c. \tag{4}$$

In Eq. (3), T_{elp} is the time that has elapsed at the moment when the rendering time is to be estimated. In Eq. (4), $TS \times C_c$ is the time that will be taken to do ray casting. As more and more changed voxels are sampled, both T_{elp} and $TS \times C_c$ increase and hence the estimated T_t(TCDVRM) increases. Once the estimated Tt(TCDVRM) is very close to the time budget, the sampling of changed voxel is stopped and the ray casting is started.

In most cases, SV is much smaller than $|P(DF_t)|$ and $D(r, s)$ is much smaller than $I_t(r, s)$. From Fig. 3, we can see that only a small fraction of randomly distributed changed voxels is enough to determine a great part of changed pixels. A small $SV \times C_p$ implies that the time to determine changed pixels is just a small ratio of a fixed time budget. Therefore, a great part of the time is spent in ray casting, which is desired in producing an image.

Table 1
The probabilities that changed pixels will be selected in Example 1

$I_t(r,s)$	$(r1, s1)$	$(r2, s2)$	$(r3, s3)$	$(r4, s4)$
$PS_t(r,s)$	100	50	10	1
SV				
1	0.6211	0.3106	0.0621	0.0062
2	0.8579	0.5260	0.1207	0.0124
3	0.9473	0.6751	0.1760	0.0186
4	0.9806	0.7779	0.2282	0.0248
5	0.9930	0.8486	0.2773	0.0311
6	0.9975	0.8971	0.3237	0.0373
7	0.9991	0.9303	0.3673	0.0435
8	0.9997	0.9529	0.4084	0.0497
9	0.9999	0.9683	0.4471	0.0559
10	0.9999	0.9787	0.4834	0.0621

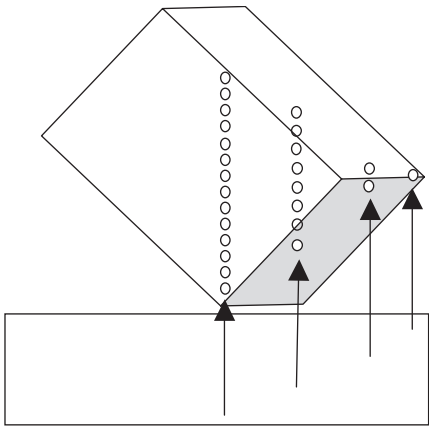

Fig. 2. The numbers of sampling points for pixels depend on the data set and view direction.

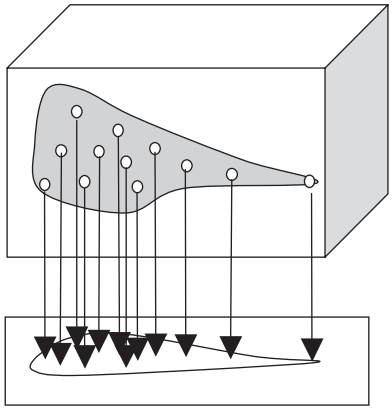

Fig. 3. A few randomly distributed changed voxels can determine a great part of changed pixels.

Only in the rare case of $|P(DF_t)|$ being very small or the time budget being long enough, all of the changed voxels are sampled before estimated $T_t(TCDVRM)$ is greater than the time budget. In such a case, SV and $D(r, s)$ equal $|P(DF_t)|$ and $I_t(r, s)$, respectively.

3.4. Algorithm of TCDVRM

The algorithm of TCDVRM is given as follows. The term C_c is determined in the first time step and is calibrated (line 19) continuously in the rest of time steps. Calibrating C_c may be avoided if the environment is very steady. It is noted here that T_{elp}, instead of C_f, C_g, and C_p, is measured in the estimation of rendering time for TCDVRM.

Algorithm. TCDVRM

/* Static phase */
1. Obtain DF_t for all t;
/* Dynamic phase */
2. Given time budget TB;
3. For time step $t = 0$\{
4. Render and Determine C_c; \}
5. For each time step $t >= 1$\{
6. Initialize elapsed time T_{elp};
7. Read differential file, update volume data, and compute gradient;
8. Let $TS = 0$; Let $D(r, s) = 0$ for all (r, s);
9. Do
10. Sample (x, y, z) from DF_t;
11. Compute $J(x, y, z)$;
12. For each $(r, s) \in J(x, y, z)$\{
13. $D(r, s)++$;
14. If $(D(r, s) == 1)$ $TS = TS + S(r, s)$;\}
15. Measure T_{elp};
16. $T_t(\text{TCDVRM}) = T_{elp} + TS \times C_c$;
17. Until ($T_t(\text{TCDVRM}) \fallingdotseq TB$) or (all (x, y, z) in DF_t are sampled)
18. Perform ray casting for each (r, s) with $D(r, s) > 0$;
19. Calibrate C_c;\}
End_of_TCDVRM

4. Experimental results

Three CFD data sets are used as test TVVD. The data sets are numerical simulations of various arrangements of jets of waste gas issuing vertically into a horizontal crossflow [19]. In the first data set, only a jet of waste gas issues into the crossflow. The data set is of the size of 81∗49∗65 voxels and has 100 time steps. In the second data set, two jets of waste gas located along the crossflow direction issue normally into a crossflow. The data set is of the size of 101∗49∗81 voxels and has 100 time steps. In the third data set, three jets of waste gas located perpendicular to the crossflow direction issue normally into a crossflow. The data set is of the size of 81∗65∗65 voxels and has 100 time steps. In all data sets, each voxel value is represented by 1 byte, which is used to represent the density of waste gas. The experiments are performed on a Pentium III 800 MHz platform with 256 MB RAM.

Fig. 4 shows the rendering time in each time step of the first data set when the time budgets are set to be unlimited, 200, and 100 ms, respectively. In the first few time steps, when the changes between consecutive time steps are very small, the rendering time may be smaller than the time budgets. After that, the rendering time almost conforms to the time budget. Fig. 4 shows that the proposed method is very effective in controlling the rendering time to meet the time budget.

Fig. 5 shows the effectiveness of TCDVRM in determining important changed pixels by sampling changed voxels. Time step 50 of the first data set is chosen as an example. In the figure, the number of all changed pixels is presented with the influence number as the horizontal axis. This histogram shows that there are totally 23 changed pixels with influence number 1, 239

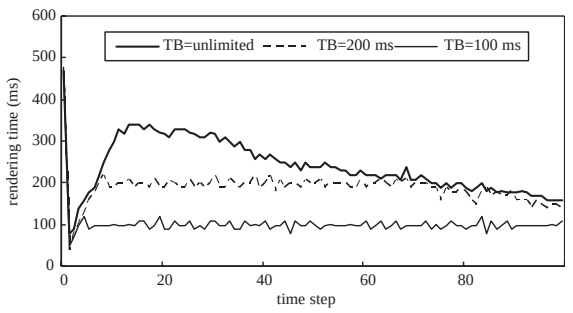

Fig. 4. The rendering time of the first data set with different time budgets.

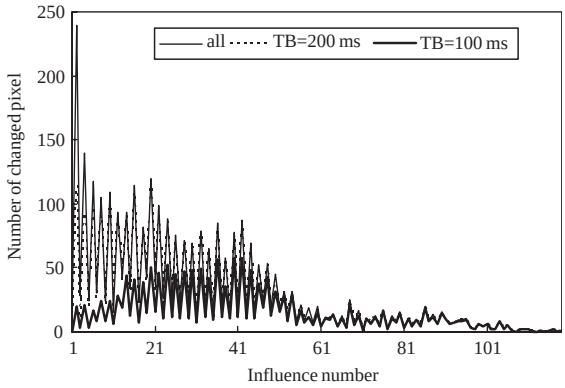

Fig. 5. Histograms of determined changed pixels with sampling changed voxels in time step 50 of the first data set.

changed pixels with influence number 2, …, and only one changed pixel with influence number 120. The histograms of changed pixels that are determined with time budgets of 200 and 100 ms, respectively, are also given. In the case that the time budget is 200 ms, one changed pixel with influence number 1, 115 changed pixels with influence number 2, …, and one changed pixel with *influence number* 120 are determined. For the time budget of 100 ms, no changed pixel with influence number 1, 20 changed pixels with influence number 2, …, and one changed pixel with influence number 120 are found. These three histograms are almost the same in the area of changed pixels with large influence numbers. In the area of changed pixels with small influence numbers, the difference between the total number of changed pixels and the number of changed pixels found with the time budget of 200 ms is pretty small. However, the difference for the time budget of 100 ms is significant. That is, when the time budget is tight, most of important changed pixels with large influence numbers are still chosen but most of unimportant changed pixels with small influence numbers are given up. With time budgets of 200 and 100 ms, the amounts of sampled changed voxels are only 15.9% and 3.4% of the total number of changed voxels, respectively. That is, our method is very effective of using few changed voxels to determine the changed pixels with large influence numbers.

Figs. 6(a)–(c) show the images of the first data set rendered with the unlimited time budget, and time budgets of 200 and 100 ms, respectively, in time step 50. In these figures, the red, green, and blue colors represent the high, medium, and low densities of waste gas. With the unlimited time budget, Fig. 6(a) is generated as an original lossless image. When the time budget is 200 ms, the image shown in Fig. 6(b) is of excellent quality. When the time budget is 100 ms, few distortions appear in some small areas of Fig. 6(c). However, the overall

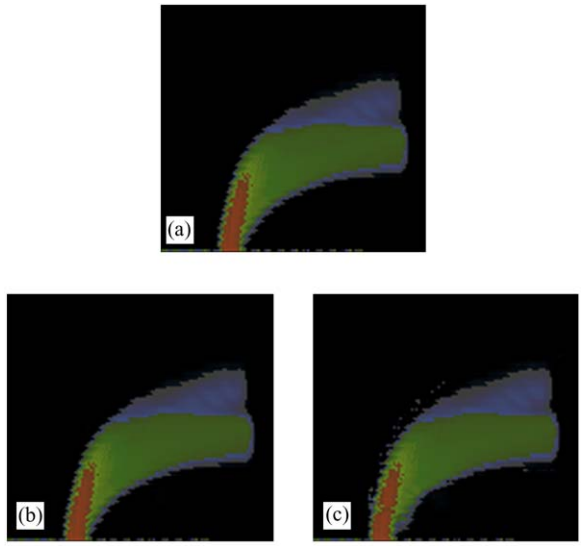

Fig. 6. Images generated for the first data set with different time budgets in time step 50: (a) unlimited, (b) 200 ms, and (c) 100 ms.

image quality is till very good. Although a lot of changed pixels with smaller influence numbers are given up as shown in Fig. 5, the artifacts due to these pixels are still acceptable.

Fig. 7 shows the rendering time in each time step of the second data set when the time budgets are set to be unlimited, 400 and 200 ms, respectively. Again, this figure shows that the proposed method is very effective in controlling the rendering time to meet the time budget. Fig. 8 shows the histograms of changed pixels in time step 50 of the second data set. Again, the number of all changed pixels is plotted with the influence number as the horizontal axis. The histograms of changed pixels that are determined with time budgets of 400 and 200 ms, respectively, are also presented. In the area of changed pixels with large influence numbers, the number of found changed pixels with time budgets of 400 and 200 ms are almost the same. It means that most of the important changed pixels with large influence numbers are determined. In the left side area, apparent gaps exist between histograms. When the time budget becomes tighter, more unimportant changed pixels are given up but most of important changed pixels are still chosen. In the cases with time budgets of 400 and 200 ms, the amounts of the sampled changed voxels are only 15.6% and 1.8% of the total changed voxels, respectively.

Figs. 9(a)–(c) show the images of the second data set rendered with the unlimited time budget, and time budgets of 400 and 200 ms, respectively, in time step 50. When the time budget is 400 ms, the generated image is of excellent quality as show in Fig. 9(b) and we almost

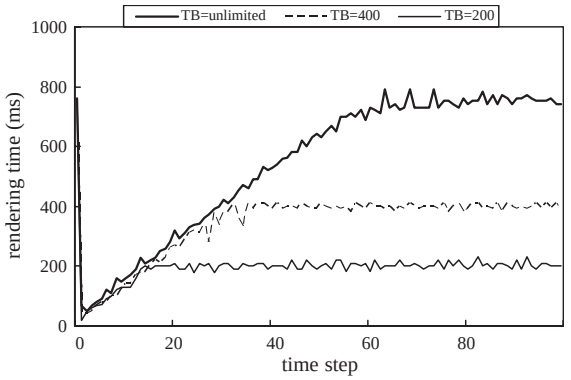

Fig. 7. The rendering time of the second data set with different time budgets.

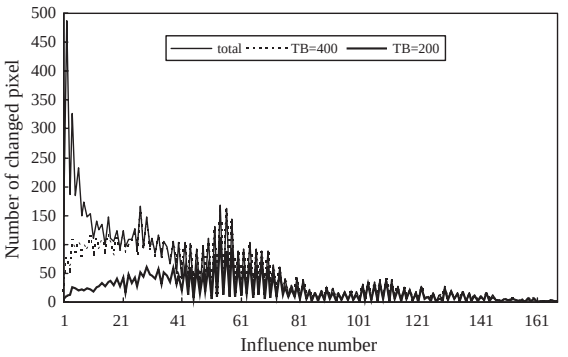

Fig. 8. Histograms of determined changed pixels with sampling changed voxels in time step 50 of the second data set.

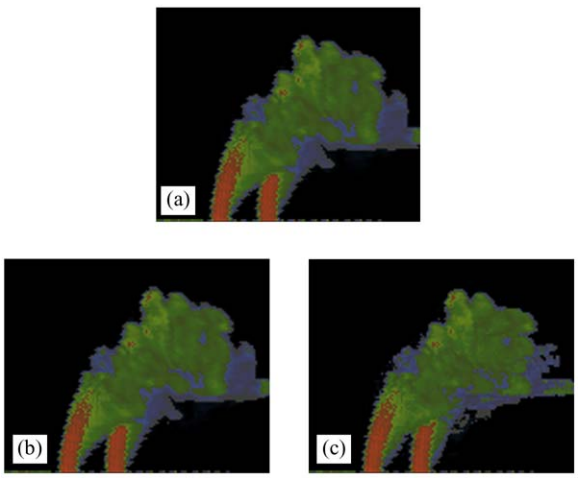

Fig. 9. Images generated for the second data set with different time budgets in time step 50: (a) unlimited, (b) 400 ms, and (c) 200 ms.

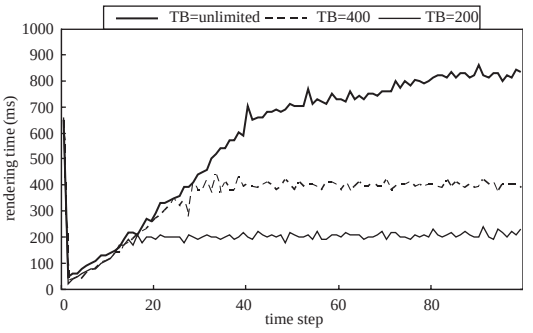

Fig. 10. The rendering time of the third data set with different time budgets.

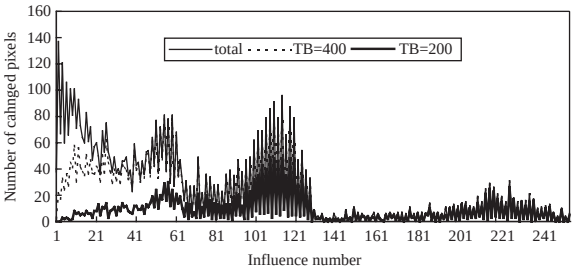

Fig. 11. Histograms of determined changed pixels with sampling changed voxels in time step 50 of the third data set.

cannot distinguish Fig. 9(a) from Fig. 9(b). When the time budget is 200 ms, some details are distorted, mostly in the green–blue area of Fig. 9(c). However, the overall shape is still preserved.

Fig. 10 shows the rendering time in each time step of the third data set when the time budgets are set to be unlimited, 400 and 200 ms, respectively. Again, this figure shows that the proposed method is very effective in controlling the rendering time to meet the time budget. Fig. 11 shows the histograms of changed pixels in time step 50 of the third data set. When the time budgets are 400 and 200 ms, the amounts of the sampled changed voxels are 8.3% and 1.1% of the total changed voxels, respectively. Figs. 10 and 11 show that our method can control the rendering time and select important changed pixels effectively.

Figs. 12(a)–(d) show the original lossless images of the third data set in time steps 25, 50, 75, and 100, respectively. Figs. 12(e)–(h) give the images of the third data set with the time budget of 200 ms in time steps 25, 50, 75, and 100, respectively. Comparing these figures and considering the constraint of 200 ms for each image frame, we can conclude that our method is good in capturing the important features of the data set within a limited time budget.

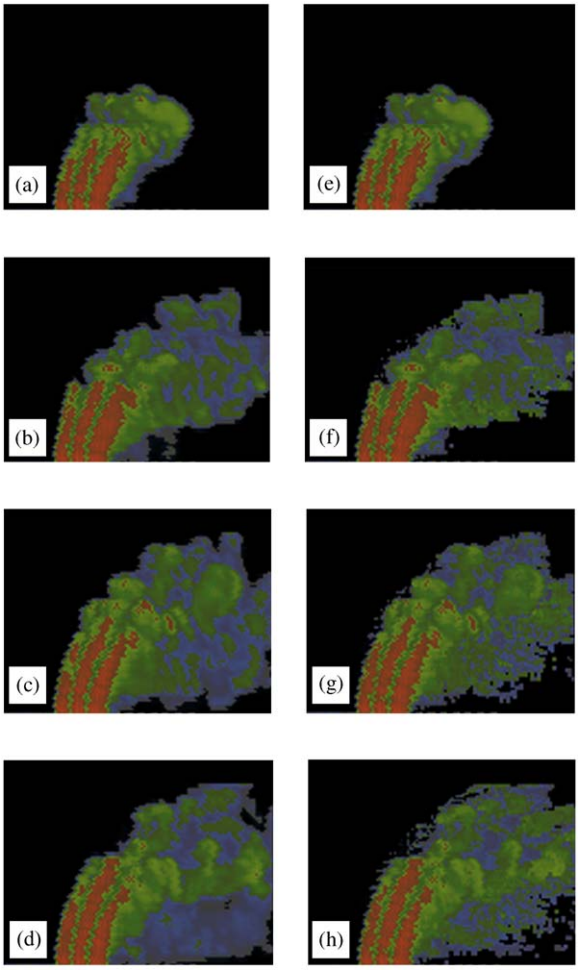

Fig. 12. Images generated for the third data set with different time budgets in time steps 25, 50, 75 and 100: (a)–(d) unlimited, (e)–(h) 200 ms.

5. Conclusions

In this paper, we present a novel approach to meet the time-critical requirement in rendering time-varying volume data. In order to control the rendering time and to get the best image quality within the time budget, we use the information of LOM, which indicates the degree of modification of changed data, as the metric to select the most important data. A novel method that integrates the sampling strategy and the estimation of rendering time is used to determine changed pixels with large influence numbers. The method for the estimation of rendering time is modified from our previous work [6]. From the experimental results on three data sets, we have the following conclusions. First, the proposed method is very effective in controlling the rendering time. Second, the LOM is good in measuring the degree of importance of changed data. When the time budget is tight, the generated images still reveal important features of the data set. Finally, the method of determining changed pixels with large influence numbers is very efficient. Only a few changed voxels are sampled to determine these changed pixels.

References

[1] Levoy M. Efficient ray tracing of volume data. ACM Transactions on Graphics 1990;9(3):245–61.
[2] Fung PF, Heng PA. Efficient volume rendering by IsoRegion leaping acceleration. The Sixth International Conference in Central Europe on Computer Graphics and Visualization '98, Plzen, Czech Republic, 1998.
[3] Lacroute P. Analysis of a parallel volume rendering system based on the shear-warp factorization. IEEE Transactions on Visualization and Computer Graphics 1996;2(3): 218–31.
[4] Shen HW, Johnson CR. Differential volume rendering: a fast volume visualization technique for flow animation. Proceeding of the Visualization '94 Conference, Washington DC, USA, 1994. p. 180–7.
[5] Anagnostouk, Atherton TJ, Waterfall AE. 4D volume rendering with shear warp factorization. Volume Visualization Graphics Symposium 2000, Salt Lake City, USA, 2000.
[6] Liao SK, Lin CF, Chung YC, Lai ZC. A differential volume rendering method with second-order difference for time-varying volume data. Journal of Visual Languages and Computing 2003;14(3):233–54.
[7] Ma KL, Shen HW. Compression and accelerated rendering of time-varying volume datasets. Workshop on Computer Graphics and Virtual Reality, 2000 International Computer Symposium, Taiwan; 2000.
[8] Sutton PM, Hansen C. Accelerated isosurface extraction in time-varying fields. IEEE Transactions on Visualization and Computer Graphics 2000;6(2):98–107.
[9] Shen HW, Chiang LJ, Ma KL. A fast volume rendering algorithm for time-varying fields using a time-space partitioning (TSP) tree. Proceedings of the Visualization '99 Conference, San Francisco, USA, 1999. p. 371–7.
[10] Chiueh TC, Ma KL. A parallel pipelined render for time-varying-volume-data. NASA/CR-97-206275, ICASE Report No. 97-70, 1997.
[11] Dobashi Y, Cingoski V, Kaneda K, Yamashita H. A fast volume rendering method for time-varying 3D scalar field visualization using orthonormal wavelets. IEEE Transactions On Magnetics 1998;34(5):3431–4.
[12] Bobbetti E, Bouvier E. Time-critical multiresolution scene rendering. Proceeding of 1999 Symposium on Volume Visualization. ACM SIGGRAPH, San Francisco, USA, 1999. p. 123–30.
[13] Funkhouser T, Sequin C. Adaptive display algorithms for interactive frame rate during visualization of virtual environment. Proceeding of SIGGRAPH, 93. ACM SIGGRAPH, New York, USA, 1993. p. 247–54.
[14] Li X, Shen HW. Time-critical multiresolution volume rendering using 3D texture mapping hardware. IEEE/ACM 2002 Symposium on Volume Visualization and Graphics, Boston, USA 2002.

[15] Cignoni P, Montani C, Puppo E, Scopigno R. Multi-resolution representation and visualization of volume data. IEEE Transactions on Visualization and Computer Graphics 1997;3(4):352–69.

[16] Li X, Shen HW. Adaptive volume rendering using fuzzy logic control. Proceeding of Joint Eurographics-IEEE TCVG symposium on visualization. Berlin: Springer; 2001.

[17] Farias R, Mitchell JSB, Silva CT, Wylie B. Time-critical rendering of irregular grids. XIII Brazilian Symposium on Computer Graphic and Image Processing, Brazil; October 17–20, 2000. p. 243–50.

[18] Porter T, Duff T. Compositing digital images. Computer Graphics 1984;18(3):253–9.

[19] Liao CB, Lu TC, Wu, MF, Feng TY. Numerical simulation of multiple vertical jets issuing into an incompressible horizontal crossflow. The Eighth National Computational Fluid Dynamics Conference, Taiwan; 2001.