Fused Visualization for Large-scale Time-varying Volume Data with Adaptive Particle-based Rendering

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
{zhao.kun, naohisas}@viz.media.kyoto-u.ac.jp
koyamada.koji.3w@kyoto-u.ac.jp

Abstract. Recently, there is a strong need for the fused visualization of different objects in many simulation fields, especially for the medical domain (e.g., the fusion of different organs). That is because it is desirable and advantageous to show the different objects and analyze the relationship between them. Nevertheless, such a simulation date is always resulted in a large-scale time-varying volume data, which make the fused visualization even more difficult. To solve this problem, we use a sorting-free rendering technique, Adaptive Particle-based Rendering (APBR), to visualize the large-scale time-varying volume data. Because this method visualizes the volume data by generating opaque particles from the original volume data and projects these particles to the image plane, the visibility sorting is not needed. This makes the fusion of different objects and handling of large-scale volume data is very easy. Moreover, our proposed APBR method can adaptively apply different particle generation process to visualize the volume data based on different viewpoints. This feature can make our system keep an interactive frame rate and also a relatively high image quality. With the APBR, we also develop a time-varying rendering into our system so that the rendering for the large-scale time-varying data also becomes possible. To verify the efficiency, we apply our APBR system to the large-scale blood flow dataset. The experimental results and the user feedbacks show that our system can fuse different objects efficiently while keeping an interactive frame rate and a good image quality, which is very meaningful in the visual analysis.

Keywords: fused visualization, large-scale volume data, time-varying visualization, particle-based rendering.

1 Introduction

Large-scale simulations can be found in many scientific fields (e.g. computational fluid dynamics, electromagnetic field simulation or ocean prediction). A good visualization result for these large-scale simulation results is always needed to clearly show detailed spatial features. However, effective visualization of such large-scale data is always difficult to achieve due to the complexity of illustrating the high-complicated

structure of the volume data. It becomes even more difficult for the time-varying simulation data, because the complex temporal features are also need to be shown clearly.

Moreover, in some recent researches, the large-scale simulations may also need to fuse different objects together to observe and analyze the relationship between these objects. This need is especially urgent for the medical simulation field [5]. For example, medical researchers may need to fuse different human organs together to observe the interaction of them. Because the medical simulation data may also be a large-scale volume data, the fused visualization for this data becomes much more difficult.

To visualize such a huge size of volume dataset and perform the fusion of different objects, the traditional visualization is always based on the extracted surfaces (polygon) or 2D images to realize the fusion of different objects [4] [5] [6] [16]. That is because different objects can also be converted into different surfaces, and the fusion of polygons and 2D images make the rendering be easy to be realized. However, a fusion of the original different objects, including volume/surface, but not the extracted surfaces, can provide a more realistic result for the positional information of these objects and can help the researchers to analyze the result more efficiently. At a result, we need a good rendering method to render the large-scale volume data and fuse different objects efficiently.

To solve this problem, we employ our particle-based rendering technique, which can handle large-scale volume datasets and easily fuse different objects, since it utilizes proxy geometries that are a set of opaque particles. The number of the particles is not in proportion to that of volume cells, but in inversely proportional to the square of the particle radius. If we determine an appropriate radius, the number of the particles can be reduced so that it can fit the GPU memory. This fitting is mandatory for interactive rendering. To develop a particle rendering algorithm, we revisited a brightness equation in the volume rendering algorithm and reconsider the definition of opacity which is usually derived from a user-specified transfer function. This leads to two approaches, object space approach and image space approach. In the former one, we define a density function of emissive opaque particles. According to the density function, we generate particles in a given volume dataset and project them onto an image plane. Since we use an opaque particle, no visibility sorting is required. In the latter approach, we regard the brightness equation as the expected value of the luminosity of a sampling point along a viewing ray, and we propose a sorting-free approach that simply controls the fragment rendering by using the evaluated opacity value to calculate a rendered image. These two approaches are called Object-space Particle-based Rendering (O-PBR) and Image-space Particle-based Rendering (I-PBR). Because we use the opaque particles which do not need the visibility sorting, the fusion of volumes and surfaces becomes very easy to be realized.

In order to get an interactive frame rate analysis while keep a good image quality, we combine both of O-PBR and I-PBR into the system. The O-PBR method can handle a large-scale time-varying data while keeping an interactive frame rate. However, the particle shape is noticeable when we enlarge the view to observe local details, which will influence the image quality (Figure 4). On the contrary, I-PBR is able to provide a very high-quality rendering result even for much enlarged view, but the renderable data size is limited. As a result, during the rendering process, our system

adaptively switches the rendering methods of O-PBR and I-PBR automatically by taking into account of the data size within the view frustum and the computer resources at all time. By doing this, we can get a smooth rendering result of the whole high-resolution data, and also observe a high-quality rendering result for some enlarged local part with interest. We call this adaptively switching rendering method as the Adaptive Particle-based Rendering (APBR). We also develop a time-varying rendering into our system so that the rendering for the large-scale time-varying data also becomes possible.

To verify the efficiency, we apply our APBR system to the large-scale blood flow dataset. The experimental results and the user feedbacks show that our system can fuse different objects efficiently while keeping an interactive frame rate and a good image quality, which is very meaningful in the visual analysis.

2 Related Work

There is a big need for the fusion of different objects in many visualization fields. That is because it is desirable and advantageous to show the different objects and analyze the relationship between them [5].

The fused visualization has a very important application in medical domain, and many approaches have been proposed. Baum et al. [5] has proposed a fusion view to fuse the different date measured from CT (computed tomography), MRI (magnetic resonance imaging), PET (positron emission tomography) and so on. The fusion result can provide a detailed analysis for the measured data. However, the measured data are 2D images, for the simulation data in 3D, the fusion cannot be realized by this system. Prckovska et al. [6] also develop a multi-field visualization framework to fuse the measured results from DTI (diffusion tensor imaging) and HARDI (high-angular resolution diffusion imaging), which is also based on the 2-dimensional image. 2D image based fusion visualization can also be found in other related work [16], which cannot fit the needs of the fused visualization for 3D volume objects and semi-transparent surfaces.

As for the fused visualization for the three-dimensional objects, how to sort the multiple objects and fuse the together comes to be a main challenge. Generally, surface-based visualization techniques, such as isosurfaces, sectional slices, and boundary faces, are useful in understanding the geometrical structure of a scalar field. Aaron et al. [7] proposed a volume ray-casting technique using peak-finding, which can integrally render the volumes and the isosurfaces extracted from the volumes, and confirmed the effectiveness of the technique by rendering several structured grid datasets. However, since this system use the ray casting rendering method, it needs to store the whole data into GPU. If the data size of the volumes and surfaces is over the GPU memory size, these data cannot be visualized. Other approaches of fused visualization for volume and opaque isosurface rendering can be found in [8] and a transparent rendering technique of semi-transparent multi-isosurfaces [9] have also been proposed. All of these methods cannot render the large-scale data with the data size over the GPU memory.

To make a fused visualization with an efficient sorting and handle a large-scale data, recently utilizing proxy geometries that are a set of opaque particles have been proposed. Koyamada et al. proposed the particle-based volume rendering (PBVR) method [10] [11] which uses tiny particles as rendering primitives. Since this method only need to project the generated opaque particles, this method does not require any sorting and applicable to large-scale data. Moreover, this method also enables volume fusion [12]. A stochastic-based particle-generation on curved surfaces has been proposed by Tanaka S. et al [13]. They also combine this approach to particle-based rendering so that the rendering for semi-transparent surfaces with this method is also possible [14]. Moreover, Hasegawa et al. show the effectiveness of this particle-based fused visualization and they demonstrate volume–volume, volume–surface, and surface-contour fusions especially for medical volume data [15]. From these related work we can see, a great advantage of the particle-based rendering is that 3D fused visualization of different volume/surface and other objects becomes possible simply by merging particles prepared for each element to be fused. However, because the particle-based rendering used in the above research, generate particles in object space (in this paper we call this as object-space particle-based rendering O-PBR), when we enlarge the view to observe some detailed local features, the particle shape becomes noticeable and the image quality becomes not fine.

As a result, in this paper, we propose an adaptive particle-based rendering method (APBR) by combining an image-space particle rendering (I-PBR) to the O-PBR to fuse the multiple volumes, surfaces. The I-PBR is originally proposed as Stochastic Projection Tetrahedra (SPT) [3]. In this method, we regard the brightness equation as the expected value of the luminosity of a sampling point along a viewing ray, and we simply controls the fragment rendering by using the evaluated opacity value to calculate a rendered image. This method is originally proposed to render the tetrahedra mesh, and in this paper, we expand it to the uniform grid and call is as the image-space particle rendering. Even though the I-PBR is not so good at handling large-scale data (generally, I-PBR cannot handle the data size over the GPU memory), the generated particle of I-PBR is in pixel scale so that we can also get a high quality rendering even for a very enlarged view. Our system can automatically switch O-PBR to I-PBR when we enlarge the view, and switch back I-PBR to O-PBR when we need to observe the entire volume so that our fused visualization system can keep a scalability to render the large scale data while keep a good rendering quality.

3 Proposed Method

In this paper, we propose an adaptive particle-based rendering (APBR) to visualize the large-scale volume data. Because this particle-based rendering use the opaque particles, the fusion of different objects and the handling of large-scale data become possible. The APBR contains two kinds of particle-based rendering: object-space particle-based rendering (O-PBR) and image-space particle-based rendering (I-PBR). During the rendering process, APBR can adaptively switch the rendering method of O-PBR and I-PBR depending on the view point to provide an interactive frame rate

and a good image quality. In this section, we will introduce the O-PBR and I-PBR separately and introduce how APBR works to switch rendering method adaptively depending on the view point. After that, the rendering process for the large-scale time-varying volume data is also shown in this section.

3.1 Object-space Particle-based Rendering (O-PBR)

Object-space particle-based rendering technique is a rendering technique based opaque particles in the object space. The original version is proposed by Koyamada et al. [1] [10] [11] [12]. In this technique, a set of opaque particles is generated from a given 3D scalar field based on a user-specified transfer function. The final image is then generated by projecting these particles onto the image plane. The particle projection does not need to be in order since the particle transparency values are not taken into account. During the projection stage, only a simple depth-order comparison is required to eliminate the occluded particles. The algorithm is listed as following.

Assuming that the volume data is a set of light-emitting cloud particles, the brightness *B* at the eye position can be solved numerically as follows [2]:

$$B_0 = \sum_{i-1}^{n} c_i \times \left(\alpha_i \prod_{j=1}^{i=1} (1 - \alpha_j) \right) \tag{1}$$

Here, a viewing ray is evenly subdivided into *n* segments, and c_i and α_i represent luminosity and opacity values at the *i*-th sampling point, which is a central point of the interval $[t_{i-1}, t_i]$. Usually, the opacity is specified through a transfer function, which is set by user. In the density emitter model, the opacity α_k in the *k*-th ray segment is defined as:

$$\alpha_k = 1 - exp\left(- \int_{t_k}^{t_{k-1}} \pi r^2 \rho(\lambda) d\lambda \right) \tag{2}$$

Here, ρ and r represent the number of particles in the unit volume and the radius of a particle, respectively. Since the Poisson distribution is assumed for the number of particles in the density emitter model, the opacity describes the possibility that more than one opaque particle exists along the ray segment. If we assume that the density function is constant in the segment, and that the ray segment length can be described as $\Delta t = t_{k-1} - t_k$, we have:

$$\alpha_k = 1 - exp(-\pi r^2 \rho_k \Delta t) \tag{3}$$

From Equation (3), the opacity can be generally expressed as:

$$\alpha_k = 1 - exp(-\pi r^2 \rho \Delta t) \tag{4}$$

As a result, the particle density used in the particle generation can be estimated using the radius, an opacity value in the user-specified transfer function, and the ray-segment length used in the ray- casting. From Equation (4), we have:

$$\rho = \frac{-\log(1-\alpha)}{\pi r^2 \Delta t} \tag{5}$$

With this particle generation function, O-PBR is composed of three parts (Figure 1):

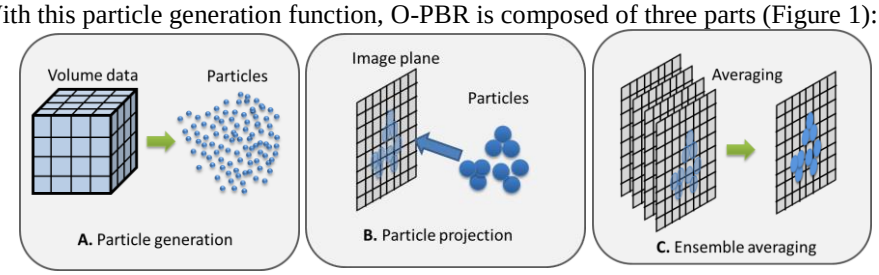
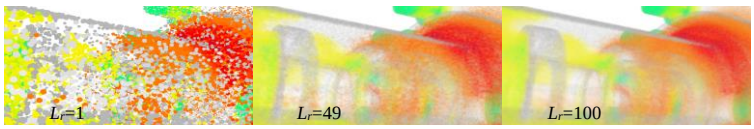
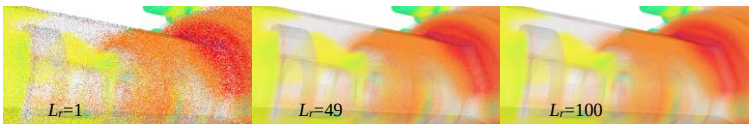


Fig. 1. The three steps of PBVR: particle generation, particle projection, ensemble average.

- **Particle generation:** A set of particles is internally generated for each volume dataset using the density distribution function (5) from a user-specified transfer function. The generation process is done in a cell-by-cell manner.
- **Particle projection:** For the second step, the generated particles are then projected onto the image plane to create a rendered image for the volume dataset. At the same time, the calculation of the particle size and the shadow processing are also performed by using the normal vector.
- **Ensemble average:** This process means that for the group of particles generated by using different random numbers, the particle projection process is performed multiple times, and the generated images are superimposed on each other to obtain the average. The multiple times is called the repetition level. In general, a larger repetition level can provide a higher image quality but needs more computational resources.



(a) O-PBR



(b) I-PBR

Fig. 2. Comparison of the image quality of the pump data in an enlarged view obtained by using O-PBR (a) and I-PBR (b). (Lr: repetition level)

Since this method generates particles in object space, we call this method as object-space particle-based rendering. With this O-PBR method, because the projection of opaque particles does not need any visibility sorting, a high-speed rendering can be realized. When we need to fuse the different volumes, we only need to combine the different particles generated from these volumes together and project them to the image plane. As a result, our O-PBR is very suitable for the fused visualization of large-scale volume data. However, the problem is that when we enlarge the view to observe the local details, the shape of the object-space particles is noticeable, which will reduce the image quality (the (b) figure in Figure 2).

3.2 Image-space Particle-based Rendering (I-PBR)

To get a high image quality with an enlarged view, we use an Image-space Particle-based Rendering (I-PBR) method to generate particles with the pixel scale during the rasterization process and deploy them on the screen (Figure 3). Compared with the O-PBR, I-PBR is able to provide a high image quality even with an enlarged view (the (b) figure in Figure 2).

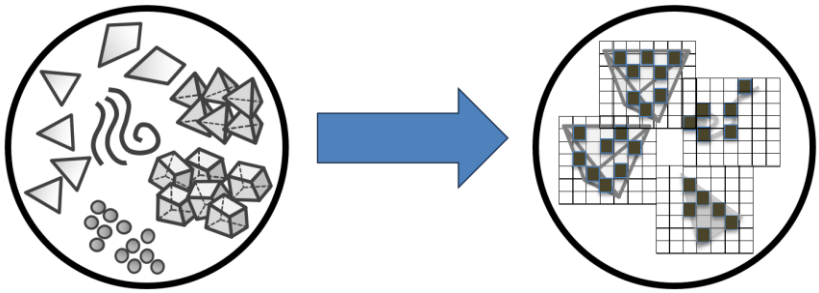


Fig. 3. The particle generation of image-space particle-based rendering.

Considering the brightness equation (1) we can construct a brightness calculation model in which there are n ray segments along a viewing ray, and the k-th particle occurs at the probability of α_k. Thus, the brightness can be regarded as the expected value of the luminosity from the ray segment:

$$B = \sum_{k=1}^{n} P_k c_k \tag{6}$$

where the possibility that the k-th luminosity c_k is equal to the brightness value can be described as follows by using the opacity value α_k:

$$P_k = \alpha_k \prod_{j=0}^{k-1} (1 - \alpha_j) \tag{7}$$

This represents an event in which there is no particle from the first to the (k-1)-th ray segment, and there is more than one particle in the k-th ray segment. In this case, the brightness B becomes c_k since opaque and emissive particles are used (Figure 4).

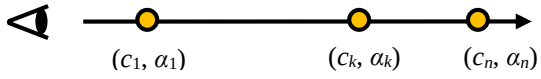
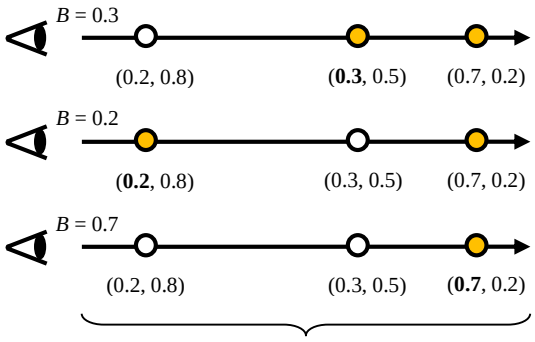


Fig. 4. Brightness calculation using luminosities and opacities.

Please note that the brightness is not contributed to by the ray segments from the k-th to the last segments since the (k-1)-th particle completely occludes these segments. This interpretation of the brightness equation suggests that a volume rendering can be approximated by repeating such events multiple times, as in the following steps:

- **Particle generation:** A pixel scale particle is generated with a probability equal to the opacity value at each ray segment along a viewing ray.
- **Particle projection:** The luminosity of the particle nearest to the viewpoint is assigned to the brightness value.
- **Ensemble average:** Steps 1 and 2 are repeated, in sequence, N times, and the resulting brightness values are accumulated to obtain the average.



Final brightness: (**0.3**+**0.2**+**0.7**) / 3 = 0.4

Fig. 5. Expected value of brightness.

Figure 5 shows that the brightness value is calculated as the average of three luminosities: 0.3, 0.2, and 0.7, in the case of N=3. In this figure, the values in parentheses represent luminosity and opacity values, respectively. In addition, the yellow circle indicates the generated particle. For example, in the first repetition, from the viewing point, the last two particles are generated, and the nearest particle whose luminosity is 0.3 is selected as the average. The multiple times N is the ensemble number. Generally, a larger ensemble number will provide a better image quality.

With this image-space particle-based rendering, visibility sorting is also not needed, which make the fused visualization be easy. Moreover, because the projected particles are in pixel scale (Figure 5) so that we can obtain a high quality rendering image with an enlarged view point (the lower figure in Figure 4). However, we need to

store the volume data in the GPU memory so that large-scale volume data with a big size is not suitable for this rendering technique.

3.3 Adaptive Particle-based Rendering (APBR)

In this section, we show how we develop an adaptive particle-based rendering to adaptively render large-scale time-varying data with switching the render of O-PBR and I-PBR by taking into account of the computer resources at all time.

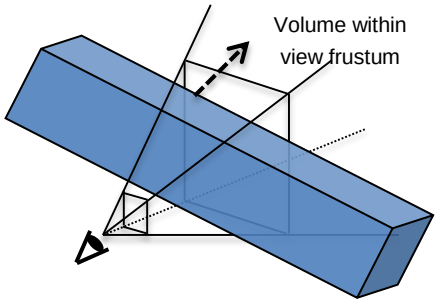


Fig. 6. The volume within the range of view frustum

Since the limitation for I-PBR is the GPU memory size, we can only use the I-PBR to visualize the part of volume within the view frustum if such a part of volume size is smaller than the available GPU size. As a result, during the rendering process, we monitor the available GPU memory and the data size in the range of the view frustum (Figure 6) at all the time.

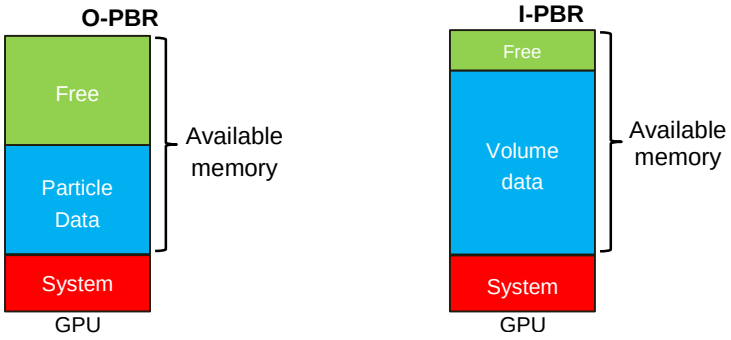


Fig. 7. Available GPU memory in O-PBR and I-PBR process.

To calculate the volume size within the view frustum, we use proxy geometry of the original volume data to check the volume with the view frustum range. In detail, we first uniformly divide the entire volume data into lots of little cubes as the proxy structure when we load the data into main memory. During the rendering process, we check the each cube that whether this cube is within the view frustum or not. Then we

calculate the segmented volume size, which consists of all the cubes within the view frustum, as the data size within view frustum.

In our system, we calculate the available GPU size as the Figure 9. We assume the GPU memory, excluding the system used, as the available memory. As shown in Figure 7, in the O-PBR process, the particle size and the free memory is the available GPU memory. In the I-PBR process, the volume data size and the free memory is the available GPU memory. We use CUDA to monitor free GPU memory, and calculate the particle data size and the volume data size in CPU before the data is transferred to GPU.

As the pre-process, we first generate particles from the original volume data. This particle data is used in the O-PBR rendering (note that, the I-PBR does not need to pre-generate particles because it generate particles in GPU using the segmented volume). During the rendering process, we first load the particle data and render it with O-PBR. Assume the data size within view frustum as S_D, and the available GPU memory as S_{GPU}, we switch the rendering method with the following rules (Figure 8):

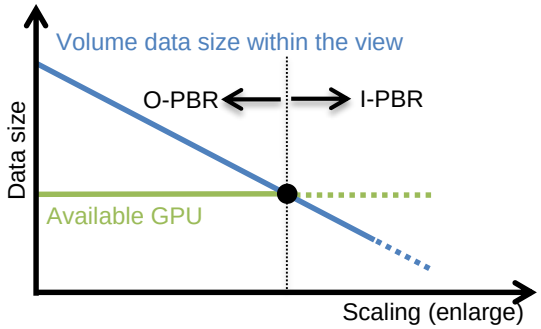


Fig. 8. The criterion to switch the renderer of O-PBR and I-PBR.

- If $S_D < S_{GPU}$,
  - Segment the volume within the view frustum,
  - Delete the particle data in GPU
  - Transfer the segmented volume to GPU and render this volume with I-PBR
- If $S_D > S_{GPU}$
  - Delete the volume data in GPU
  - Transfer the particle data to GPU and render it with O-PBR

With this adaptive visualization system, when we visualize the large-scale volume data, we can first get a rendering image of the whole data with O-PBR and then visualize the high-resolution details on an enlarged scale with I-PBR.

3.4 Animation Rendering

With the above adaptive particle-based rendering method, the visualization for the large-scale time-varying volume data also becomes simple to be realized. As mentioned before, the O-PBR is very suitable to handle the large size volume data, so we use the O-PBR to perform the animation rendering. As the preprocess, we first generate the object-space particle data from the original volume data (Figure 9). The generated particle data size can be much smaller than the original volume data so that the time-varying rendering becomes possible. The generated time-varying particle data is rendered with O-PBR time step by time step as an animation, which can clearly show the temporal variations of the time-varying data. When we stop the animation at some time step, our system load the original volume data at this time step and render this time step with our APBR (Figure 9). As a result, the temporal features can be shown clearly with O-PBR, and the detailed analysis for some time step can also be realized.

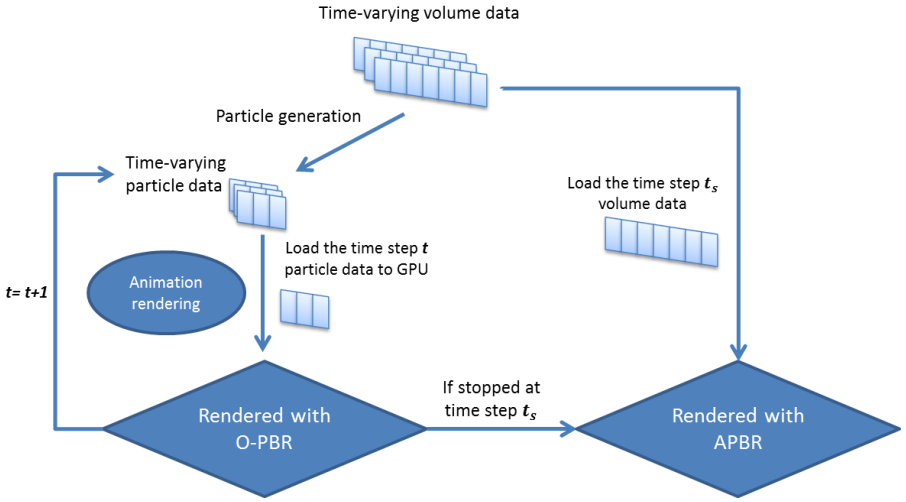


Fig. 9. The rendering process for the time-varying data.

4 Experiments and Results

In this section, we apply our system to the large-scale blood flow datasets. The blood flow dataset was obtained from a numerical simulation performed on the K computer by Sugiyama et al. [4]. The output data have a high resolution of 3,072×640×640. The dataset contains volume data for red blood cells and platelets. The entire dataset contains 100 time steps. For each time step, the data size reaches approximately 20 GB for the red blood cell volume and platelet volume. This experiment is conducted with an Intel Core i7-2820QM CPU (2.3 GHz), an NVidia GeForce GTX580M 2 GB GPU, and 16 GB of system memory. The operating system is Ubuntu 12.04 LTS. In the experiment, we first convert the data type of double to unsigned char. We fuse the red blood cell volume and platelet volume and measure the switching time of the

renderers, show the rendering image, and obtain the user feedbacks to confirm the effectiveness of our system.

4.1 Visualization Results

With our proposed system, we are able to fuse the red blood cell volume and platelet volume together in the visualization result. As the user interface, we have a time slider to help user choose the interested time step. We also show the details of the details for the free GPU memory; data size of the volume within the view frustum and the data size of the pre-generated particle data size to monitor the available GPU memory (see details in Figure 7). Figure 10 shows the animation rendering result for 9 time step (from time step 60 to time step 68). This rendering result can clearly show the temporal variations of the red blood cells and platelets and the interaction of them.

Then we choose one time step (here it is the time step 8) to perform the detailed analysis. Figure 11 shows the automatically switching from O-PBR to I-PBR. With the O-PBR, we can observe the entire state of the blood stream Then, we enlarge the view and our system automatically switches the renderer to I-PBR. We can get a high quality rendering to analyze the local features for the enlarged view. The detailed distribution of the red blood cell and the platelets are very clear to be observed. Furthermore, when we zoom out, our system automatically switches the render to O-PBR. During the rendering we use a repetition level (ensemble number) of 15 for O-PBR , and a repetition level of 50 for I-PBR to keep a frame rate larger than 10 while maintain a good image quality. We also measured the switching time between the O-PBR and I-PBR. When the renderer is switched from O-PBR to I-PBR, it cost about 2.16s. But for the switching from I-PBR to O-PBR, it only costs about 6.44×10^{-2}s.

4.2 User Feedbacks

We also show this system to some domain experts and get user feedbacks from them. The main feedbacks are summarized as following:

1. The interactive visualization for such a large-scale dataset is possible on a normal computer. This is very helpful for the visual analytics. In the previous work, the visualization for this data need to be performed with a super computer.
2. The fused visualization for the red blood cell volume and platelet volume is very helpful to know the he positional relation of them. This is very important to analyze some blood diseases such as deformable vesicle problem.
3. The animation rendering can show the time-varying changes very clearly. This is very meaningful to know the dynamic motions and interactions.
4. The whole view for the entire data and detailed analysis for the enlarged local part is provided. Detailed and high precise visual analysis becomes possible.
5. The image quality is good enough to perform the visual analysis.
6. During the switching of renderer from O-PBR to I-PBR, it costs a little long time.

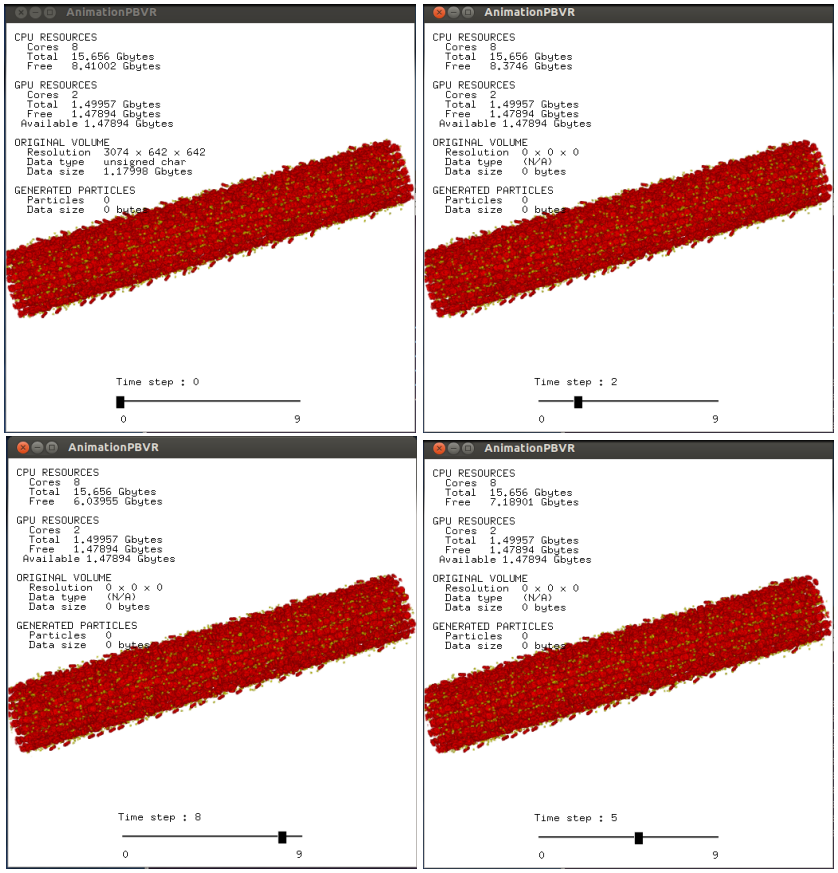


Fig. 10. This animation rendering for different time step.

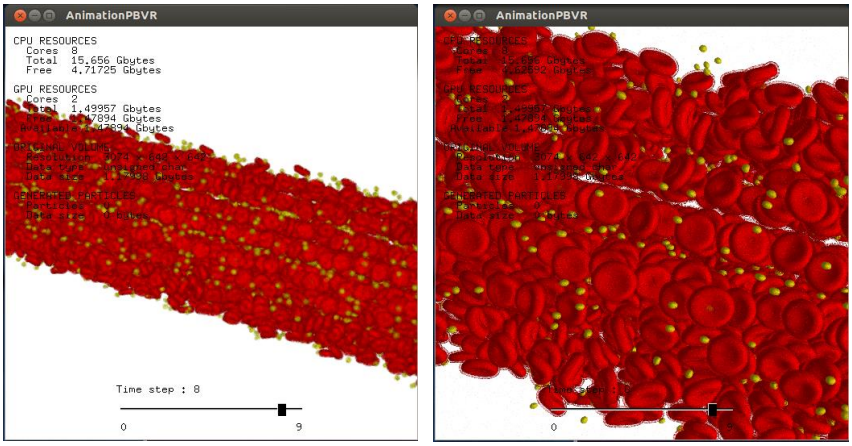


Fig. 11. The rendering result with APBR. The above figure is the rendering result with O-PBR, the bottom figure is the rendering result with automatically switched I-PBR.

5 Discussion & Conclusion

The above experimental results show our system can fuse different objects efficiently while keep an interactive frame rate and a good image quality for the visualization of large-scale time-varying volume datasets. With this system, the interactive visual analytics is possible with even a normal computer. The user feedbacks of 1, 2, 3, 4 and 5 also confirm that the proposed system is very efficient to perform the visual analytics. The image quality of both O-PBR for the global view point and I-PBR for an enlarged local view point is also good enough to perform the analysis.

However, there are also some disadvantages of our system. As commented in the user feedbacks 6, the switching from O-PBR to I-PBR cost much time (2.16s). The main reason for this is that the volume segmentation process is needed, and such a process can cost much time. Compared with this, the switching from I-PBR to O-PBR is fast because such process is not needed. Such long switching problem can give a bad influence for the interactivity if user perform many scaling operation. In our future work, we would like to develop some efficient volume segmentation method to accelerate the switching process from O-PBR to I-PBR

In this paper, a fused visualization system to visualize the large-scale blood flow datasets with adaptive particle-based rendering (APBR) has been proposed. Because the APBR does not need any visibility sorting, it can handle a large-scale volume datasets, and the fusion of different objects is also easy to be realized. To get an interactive frame rate analysis while keep a good image quality, APBR is combined with two rendering methods: O-PBR and I-PBR. O-PBR can render the large-scale data at a high speed and a high quality for the entire data, but the image quality is not so fine when we enlarge the view to observe some interested local details. I-PBR is able to provide a very high-quality rendering result even for much enlarged view, but the renderable data size is limited. In the proposed system, to achieve an interactive frame-rate rendering while keep a good image quality, we developed an adaptive particle-based rendering which to switch the above two rendering methods automatically by taking into account of the data size within the view frustum and computer resources. Such adaptive particle-based rendering can provide a smooth rendering result of the whole high-resolution data, and also observe a high-quality rendering result for some enlarged local part with interest. We also developed a time-varying rendering into our system so that the rendering for the large-scale time-varying data also becomes possible. The experimental results show that our system is very efficient to visualize the large-scale time-varying volume datasets.

ACKNOWLEDGMENTS

This work was supported by Japan Society for the Promotion of Science (JSPS) KAKENHI Grants-in-Aid for JSPS Fellows (Grant Number 26・837), and was partially supported by the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Grat-in-Aid for Data Integration and Analysis System (DIAS), Grant-in-Aid for Research Programs on Climate Change Adaptation (RECCA) and by the Japan Science and Technology Agency (JST), A-STEP project ("The research and development of fusion visualization technology", AS2415031H).

REFERENCES

1. 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 , pp. 905-915, 2010.
2. P. Sabella, “A Rendering Algorithm for Visualizing 3D Scalar Field”, Computer Graphics, Vol.22, No.4, pp.51-58, 1988.
3. N. Sakamoto, T. Kawamura, H. Kuwano, K. Koyamada, “Sorting-free Pre-Intergrated Projected Tetrahedra”, In Proceedings of the 2009 Workshop on Ultrascale Visualization (UltraVis2009), pp.11-18, 2009.
4. K. Sugiyama, Y. Kawashima, S. Noda, S. Ii, H. Koyama, S. Takagi, Y. Matsumoto and R. Himeno, “Massively parallel computing of novel fluid-structure interaction solver on the K computer”, Proc. of High Performance Computing Symposium 2013, (Tokyo, Japan, Jan. 2013), IPSJ-HPCS2013050, 2013
5. K. G. Baum, M. Helguera and A. Krol, “Fusion Viewer: A New Tool for Fusion and Visualization of Multimodal Medical Data Sets”, J Digit Imaging, Vol. 21(Suppl 1), pp. 59–68, 2008.
6. V. Prckovska; T. H. J. M. Peeters, , M. Van Almsick, B. ter Haar Romeny, A. Vilanova i Bartroli, "Fused DTI/HARDI Visualization," Visualization and Computer Graphics, IEEE Transactions on , Vol.17, No.10, pp.1407-1419, 2011.
7. A. Knoll, Y. Hijazi, R.Westerteiger, M. Schott, C. Hansen and H. Hagen, “Volume Ray Casting with Peak Finding and Differential Sampling”, IEEE Transactions on Visualization and Computer Graphics, Vol.15, No. 6, pp 1571-1588, 2009.
8. R. Marroquim, A. Maximo, R. Farias, and C. Esperança, “Volume and Isosurface Rendering with GPU-Accelerated Cell Projection”, Computer Graphics Forum, Vol.27, No.1, pp.24-35, 2008.
9. P. Kipfer and R. Westermann, “GPU Construction and Transparent Rendering of Iso-Surfaces”, In Proc. of Vision, Modeling and Visualization 2005, pp.241-248, 2005.
10. K. Koyamada, N. Sakamoto, S. Tanaka, “A particle modeling for rendering irregular volumes”, Proc. Int. Conf. Computer Modeling and Simulation (UKSIM 2008), Cambridge, pp. 372–377, 2008.
11. N. Sakamoto, T. Kawamura, K. Koyamada, “Improvement of particle-based volume rendering for visualizing irregular volume data sets”, Comput. Graph. Vol. 34, No. 1, pp. 34-42, 2010.
12. N. Sakamoto, K. Koyamada, A. Saito, A. Kimura, S. Tanaka, “Multi-volume rendering using particle fusion”, IEEE VGTC Pacific Visualization Symp. 2008, Kyoto, 2008.
13. S. Tanaka, A. Morisaki, S. Nakata, Y. Fukuda, H. Yamamoto, “Sampling implicit sur- faces based on stochastic differential equations with converging constraint”, Comput. Graph., Vol. 24, No. 3, pp. 419-431, 2000.
14. S. Tanaka, K. Hasegawa, Y. Shimokubo, T. Kaneko, T. Kawamura, S. Nakata, S. Ojima, N. Sakamoto, H. Tanaka, K. Koyamada, “Particle-based transparent rendering of implicit surfaces and its application to fused visualization”, Short paper Proc. EuroVis 2012, Vienna, pp. 5-29, 2012.
15. K. Hasegawa, S. Ojima, Y.i Shimokubo, S. Nakata, K. Hachimura and S. Tanaka, “Particle-based Transparent Fused Visualization Applied to Medical Volume Data”, International Journal of Modeling, Simulation, and Scientific Computing, Vol. 4, Supp01, pp. 1341003, 2013
16. N. Kocer, O. Kizilkilic, D. Babic, D. Ruijters, and C. Islak, “Fused magnetic resonance angiography and 2D fluoroscopic visualization for endovascular intracranial neuronavigation”, Journal of Neurosurgery,Vol. 118, No. 5, pp. 1000-1002, 2013.