Particle-Based Volume Rendering of
Unstructured Volume Data

Takuma KAWAMURA¹) Jorji NONAKA¹) Naohisa SAKAMOTO²(3) Koji KOYAMADA²)

¹) Graduate School of Engineering, Kyoto University
²) Center for the ...., Kyoto University
³) KGT inc.

1. INTRODUCTION

Numerical simulations based on Finite Element Methods such as Computational Fluid Dynamics and Computational Structural Mechanics commonly use unstructured grid data in the form of mesh structures, such as tetrahedral meshes. Scientific visualization is a proven effective technique for extracting meaningful information from these datasets, and an efficient visualization method for unstructured grid data is currently one of the major research goals in the scientific visualization community.

Volume visualization of unstructured grid data is considerably more complicated than that of regular grid data since the sampling and compositing processes, which must be carried out in visibility order, are not straightforward. The sorting into visibility order has always prevented efficient volume rendering of unstructured volume data.

Our approach for volume rendering of unstructured volume data composed of tetrahedral cell elements is based on particle-based volume rendering [2], which doesn’t require alpha blending composition in visibility order. The particle-based volume rendering utilizes stochastically generated tiny, non-transparent particles as the rendering primitives and projects particles to image plane.

Since the particles are non-transparent, alpha blending composition in visibility order is not required. We defined the regularly placed sampling points inside the tetrahedral cells as being the candidate points for particle generation. We applied the voxelization technique for sampling points at regular intervals inside the tetrahedral cells. The particle generation process is based on a density distribution function based on the transparency values (α values) defined by user-specified transfer function. These generated particles are projected onto the image plane in order to obtain the final image.

Splatting based volume rendering method generally projects large semi-transparent elements onto the image screen affecting a set of pixels and the contribution to each of the involved pixels are determined by the projection footprint. On the other hand, particle-based volume rendering approach projects tiny, particles much smaller than the pixel size. These particles are colored according to the parameter values obtained from the user-specified transfer function. The final pixel value is then obtained by calculating the contribution of these tiny particles projected onto the corresponding pixel location by using sub-pixel processing. The degree of transparency can be controlled by changing the sub-pixel size.

In the following sections, these processes will be detailed described in turn.

2. RELATED WORK

Various approaches for volume rendering of unstructured volume data have been proposed, including Ray-casting [5], Splatting [6][7], and Software and Hardware-Assisted Cell Projection [1][8].

Ray-casting is known to generate better volume rendering results; however, when handling unstructured volume data, point location during the sampling stage is a laborious task.

Although Splatting and Cell Projection-based approaches are considered better with regards to rendering speed, the visibility ordering still remains a bottleneck.

Most of the cell projection algorithms are based

¹) e-mail: kawamura@viz.media.kyoto-u.ac.jp
on the Projected Tetrahedra (PT) algorithm [1]. In the PT algorithm, each tetrahedral cell is converted into triangles and projected onto the screen in visibility order from back to front in order to generate a final semi-transparent image. This process should be performed each time a new viewing position is specified.

Splatting-based algorithms [6][7][9] splat semi-transparent elements onto the image plane, an alpha blending composition in visibility order is required. Our proposed volume rendering technique is closely related to the Splatting algorithm; however our technique uses much smaller non-transparent light-emitting particles which are stochastically generated.

3. VOXELIZATION OF TETRAHEDRAL CELLS

In this paper, we extended the voxelization approach first proposed for hexahedral cells called “Two-Pass Rasterization” [11]. In this technique, two steps are required: projection onto a 2D plane and a scan conversion along the depth direction. Therefore, there is no need to enclose the tetrahedral cells using a bounding box, such as in [4], nor to use two projection stages, as in [10], where the front and back faces are projected onto front and back frame buffers, respectively. In the following sections, the voxelization procedure of tetrahedral cell elements will be detailed.

3.1. Two-Pass Rasterization Algorithm

The two-pass rasterization algorithm for tetrahedral cell elements works as follows:

Step 1. A tetrahedral cell is projected onto an XY plane (Figure 1). From the four projected cell nodes we determine the minimum and maximum coordinates, \( y_{min} \) and \( y_{max} \), in the y-axis direction. From these values we calculate the initial \( (l_{y1}) \) and final \( (l_{yI}) \) scanlines in the y-axis direction. If there are no Cartesian grid points existing between these two scanlines, the rasterization process is finished. Otherwise, the following steps are executed starting from the initial scanline \( (l_{y1}) \). The current active scanline is defined as \( l_y \).

Step 2. Verification of the points of intersection between the current scanline \( (l_y) \) and the projected edges of the tetrahedral cell is carried out (Figure 2). The intersected edges give us the minimum and maximum coordinates, \( x_{min} \) and \( x_{max} \), in the x-axis direction. Similar to the previous step, from these values we calculate the initial \( (l_{x1}) \) and final \( (l_{xI}) \) scanlines in the x-axis direction. If there are no Cartesian grid points existing between these two scanlines, the rasterization process goes back to Step 1 in order to process the subsequent scanline \( (l_{y+i}) \) in the y-axis direction.
Step 3. Two tetrahedral faces (triangles) which possess two or more active edges are selected. The intersection points, \( Z_{\text{min}} \) and \( Z_{\text{max}} \), on these triangles defined by the current scanlines \( l_{\text{in}} \) and \( l_{\text{out}} \) are calculated. From the intersection points we calculate the initial \( (l_{x1}) \) and final \( (l_{x2}) \) scanlines in the z-axis direction (Figure 3). If there are no Cartesian grid points existing between these two scanlines, the rasterization process goes back to Step 2 in order to process the subsequent scanline \( (l_{n+1}) \) in the x-axis direction.

Step 4. Defining the active triangles as front \( (\Delta_{\text{front}}) \) and back \( (\Delta_{\text{back}}) \), the next procedure is to interpolate the data between the initial \( (l_{x1}) \) and final \( (l_{x2}) \) scanlines in the z-axis direction. This data interpolation is executed by linearly interpolating the data values of \( Z_{\text{min}} \) and \( Z_{\text{max}} \). The values of \( Z_{\text{min}} \) and \( Z_{\text{max}} \) are obtained by bilinearly interpolating the data values present at the nodes of the triangles. After processing all grid points, the process goes back to Step 2 in order to process the subsequent scanline \( (l_{n+1}) \) in the x-axis direction.

4. PARTICLE-BASED VOLUME RENDERING

The illumination model used in the particle-based volume rendering method is more closely related to the Sabella’s varying density emitter model [3], which uses much smaller particles than traditional splatting algorithms and assumes that the particles are fully opaque. These small particles are treated as though emit light themselves and compose a cloud-like material. Considering that the particles emit light with no scattering, the brightness equation can be described as shown in equation 1.

\[
B = \int_{t_0}^{t_0} \rho(t) \times c(t) \times \exp\left( - \int_{t}^{t_0} \rho(\lambda) \, d\lambda \right) \, dt \tag{1}
\]

The variable \( \rho \) denotes the number of particles per unit length along the viewing ray and \( c \) denotes the luminosity per unit area of a particle projection. Generally, the brightness \( B \) and the luminosity \( c \) have three independent components representing the red, green and blue colors. The parameters \( t_0 \) and \( t_\text{f} \) represent respectively the shortest and farthest distances from the viewpoint along a viewing ray.

Traditional volume rendering methods are based on an approximation model where a set of small particles present in a unit length is treated as the smallest entity. The density of the particles present in a unit length determines the degree of transparency. Therefore, the smallest entity treated by traditional volume rendering methods is a semi-transparent entity. In addition, the image composition of these semi-transparent entities via over operator [12] must be in the visibility order. Particle-based volume rendering approach, on the other hand, treats the small particles as the smallest entities while striving to emulate as closely as possible the Sabella’s varying density emitter model. Since, the particles themselves are non-transparent entities an image composition in the visibility order is unnecessary.

4.1. Particle Density

Since computational systems cannot solve the integral equation in its current form, approximations such as using the Riemann sum is usually applied. It is possible to discretely solve equation 1 by subdividing it into \( n \) equal sub-domains and treating the luminosity as being constant. In this case, the brightness \( B_k \) for the \( k \)-th segment can be described as shown in equation 2. From this equation we can define an opacity value \( \alpha_k \) in the \( k \)-th sub-domain:

\[
B_k = \left( \prod_{r=0}^{k-2} \exp\left( - \int_{t_i}^{t_{i+1}} \rho(\lambda) \, d\lambda \right) \right) \times c_k \times \left( 1 - \exp\left(- \int_{t_i}^{t_{i+1}} \rho(\lambda) \, d\lambda \right) \right) \tag{2}
\]

\[
\alpha_k = 1 - \exp\left(- \int_{t_i}^{t_{i+1}} \rho(\lambda) \, d\lambda \right) \approx 1 - \exp\left( - \rho_k \times \Delta t \right) \tag{3}
\]

where \( \Delta t \) is the length of the sub-domain and \( \rho_k \) is the representative density. In the volume rendering process, these opacity values are usually obtained from scalar values \( S \). However, in the particle-based volume rendering approach the particle densities implicitly relate to scalar values as shown in equations 4.

\[
\alpha(S(x)) = 1 - \exp\left( - \rho \times \Delta t \right) \tag{4}
\]

where \( x \) denotes the position along a viewing ray. The particle density can be determined from an opacity value converted from a scalar value using a
user-specified transfer function.

4.2. Particle generation using Hit-and-Miss Method

The particle generation problem based on a density distribution function can be treated as a random number generation problem based on probability density function. By considering the particle generation position \( x \) as a stochastic variable (random number) and the density function \( \rho(x) \) as the probability density function for \( x \), we can determine \( x \) according to \( \rho(x) \). In other words, it is possible to decide whether or not to generate particles at any position \( x \) based on the density function \( \rho(x) \).

We used the Hit-and-Miss method to generate these random numbers following a given continuous probability density function. The Hit-and-Miss method can efficiently generate particles simply by comparing the density distribution of particles with the generated random numbers. The processing procedure of the Hit-and-Miss method is as follows:

**Step 1.** Generate a random number \( Y \) according a given density function \( g(x) \), to which the domain of \( f(x) \) is equivalent.

**Step 2.** Generate a random number \( R \) between the range \([0, 1]\).

**Step 3.** Verify if the value of \( R \) is less than \( f(Y)/\{cg(Y)\} \). If so, accept \( Y \) such that \( x = Y \). Otherwise, reject \( Y \) and go back to Step 1. Here, \( c \) is a constant that satisfies \( c > f(x)/g(x) \).

In the proposed volume rendering method for tetrahedral cell based unstructured volume data, the particles are assumed to be generated at regularly distributed intervals inside the tetrahedral cells by using voxelization method. This particle generation process is executed based on a given density distribution function \( \rho \) and the Hit-and-Miss method. The random number \( Y \), generated in Step 1 at the positions obtained by voxelization, determines whether a particle will be generated or not at each of the voxelized positions. In the proposed method, for simplification, it is assumed that \( g(x) = 1 \) and \( c = 1 \). Therefore, the particle generation at a given voxelized position \( x \) is decided by simply comparing the density \( \rho \) with the generated random number \( R \) as follows:

**Step 1.** Generate a random number \( R \) between the range \([0, 1]\).

**Step 2.** If \( R \leq \rho(x) \), then generate a particle at \( x \). Otherwise, do not generate a particle at position \( x \).

5. EXPERIMENTAL RESULT AND DISCUSSION

We implemented a particle-based volume rendering application for tetrahedral cell based unstructured volume data using the KVS (Kyoto Visualization System) library [13]. The KVS library is a multi-platform, open-source C++ library developed in our laboratory for developing numerical simulation and scientific visualization applications. In this first attempt at implementing this proposed pure software based volume rendering application, we used Cygwin, a Linux-like environment running on the Windows operating system, as the platform for software development. We used a commodity PC (which CPU is Pentium Core Duo 3.2 GHz, and Memory: 2GB RAM, OS: Windows XP) and two tetrahedral cell-based unstructured volume datasets for preliminary evaluation of the implemented volume rendering technique. Figure 4 shows the rendering result of
unstructured volume data using particle-based volume rendering. One of the volume datasets was composed of only one tetrahedron and was named “Tetrahedron” (Figure 4(a)). The other was a simulation result of internal blood flow in an arch of the aorta and was named “Aorta” (Figure 4(b)). It is composed of 1,386,882 tetrahedra.

The required processing time for the voxelization and particle generation are shown in figures 5. In the case of the Tetrahedron data, the voxelization time was small since only one tetrahedron needed to be processed, whereas more than one million tetrahedra were required to be processed in the Aorta data. The required computational time for the particle generation process was similar in both utilized data. This was because the quantity of candidate points for particle generation was similar in both cases as we can confirm in table 1. From this table we can verify that the sampled points within a virtual volume data enclosing the unstructured data show about 16.83% in the case of Tetrahedron data using the resolution sizes shown in table 1. On the other hand, it shows about 14.76% in the case of Aorta data. We can easily verify from these values that traditional voxelization approach for visualizing unstructured volume data is inefficient since a considerably loss on memory resource usage is generated. It is worth noting that the sampling points for the Tetrahedron data corresponds exactly to the sampling points inside a single tetrahedral cell. However, in the case of Aorta data these sampling points correspond to the sampling executed in a more than one million tetrahedral cells. Therefore, for sampling resolutions lower than 128³, most of the tetrahedral cells will not have any sampling point inside them thus an undesirable loss on information is expected. Although we can obtain the number of sampled points larger than that of unstructured volume data using a higher resolution sampling, it is important that the sampling interval should be at least equal to the size of the smallest tetrahedron in order to avoid any data loss. In the Aorta data, the sampling rate equivalent to an enclosing volume data of size 256³ was considered sufficient for reconstructing the entire volume data with a minimum data loss.

Visualization results using different voxelization sampling resolutions are shown in figures 6. As mentioned in section 4, the particles sizes are smaller than a pixel size therefore it is required to generate sufficient quantity of particles for projecting at least one particle for every active pixel space in order to avoid rendering artifacts. From figures 6 we can easily observe rendering artifacts when the sampling resolution is lower than the screen resolution, i.e. there is no sufficient quantity of particles to fulfill the entire active pixel space with at least one particle per pixel in the corresponding pixels area of the rendering object. By modifying the pixel size, i.e. the maximum allowed quantity of particles per pixel, a semi-transparency effect can be obtained. This is because the exponential term of the rendering equation (Equation 1) which corresponds to the probability of the particles to achieve the viewpoint

<table>
<thead>
<tr>
<th>Table 2: Regularly distributed sampling points</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Tetrahedron” data</td>
</tr>
<tr>
<td>Resolution</td>
</tr>
<tr>
<td>128x128x128</td>
</tr>
<tr>
<td>256x256x256</td>
</tr>
<tr>
<td>“Aorta” data</td>
</tr>
<tr>
<td>Resolution</td>
</tr>
<tr>
<td>128x128x128</td>
</tr>
<tr>
<td>256x256x256</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2: Rendering speed (FPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Tetrahedron” data</td>
</tr>
<tr>
<td>Resolution</td>
</tr>
<tr>
<td>128x128x128</td>
</tr>
<tr>
<td>256x256x256</td>
</tr>
<tr>
<td>“Aorta” data</td>
</tr>
<tr>
<td>Resolution</td>
</tr>
<tr>
<td>128x128x128</td>
</tr>
<tr>
<td>256x256x256</td>
</tr>
</tbody>
</table>

Fig 6: Visualization results for different sampling resolutions on a screen size of 256x256
proportionately increases. Figure 7 shows two different pixel sizes and their visualization results. In figure 7(a) which allows only one particle per pixel, only the closest particle becomes the pixel value thus resulting in an opaque visualization result. On the other hand, figure 7(b) shows an example where each pixel can be projected from a maximum of nine particles. In this case, we can easily perceive the semi-transparent effect. From the same figure, we can also perceive undesired rendering artifacts which occur due to the regular interval sampling. An investigation to minimize or eliminate this problem is left as a future work.

6. CONCLUSION

In this paper, we presented an efficient particle-based volume rendering method for tetrahedral cell-based unstructured volume data. Regularly distributed sampling inside the tetrahedral meshes was used, together with tiny particle generation using the Hit-and-Miss method. Since the generated particles are solid non-transparent entities, traditional alpha blending operation in visibility order, front-to-back or back-to-front, is not required. Only a depth comparison for selecting the non-occluded particles is required after the particle projection, thus eliminating the computationally costly sorting process for visibility ordering required in traditional volume rendering methods. The elimination of sorting greatly facilitates distributed and parallel processing since the problem of data dependency is minimized. Although our proposed algorithm proved efficient, particle generation only on regularly sampled points can produce undesirable rendering artifacts depending on the view point. In future work, we plan to use a more flexible particle generation approach, such as one based on the Metropolis technique. In addition, we also plan to parallelize the proposed volume rendering algorithm and to evaluate its performance on High Performance Computing systems.

REFERENCE