Volume Rendering Method for Spatial Seismic Data Based on GPU

DUAN Zhongxiang
College of Geophysics and Information Engineering,
China University of Petroleum
Beijing Key Lab of Data Mining for Petroleum Data
PanPass Institute of Digital Identification Management
and Internet of Things
Beijing, 102249, China
e-mail:duanzx1987@foxmail.com

LI Guohe
College of Geophysics and Information Engineering,
China University of Petroleum
Beijing Key Lab of Data Mining for Petroleum Data
PanPass Institute of Digital Identification Management
and Internet of Things
Beijing, 102249, China
e-mail:lgh102200@sina.com

***Abstract*—Volume rendering is one of the focuses in the research and application of computing visualization. On basis of the spatial data volume formally defined，principles and methods are introduced on the division of volume data, computation of resampling and composition of image in the ray casting algorithm. By resampling and compositing with shader, the algorithm is successfully improved in performance by GPU. The application of the algorithm in seismic interpretation is implemented for visualization of spatial seismic data in gray and pseudo-color and a transfer function is designed to highlight the characteristics of stratums in seismic data field, overcoming limitations in the visualization of profiles, slices and surface rendering of seismic data.**

Keywords- ray casting algorithm; transfer function; GPU accelerating; seismic interpretation

I. INTRODUCTION

Conventional methods of seismic data visualization mainly include profile , slice and surface rendering, trying to explain the entire data field by flat or curved surface rendered with part of the seismic data volume, whose images are isolated and one-sided, unable to reflect the overall trend of the data[1]~[3].The image of volume rendering techniques is the contribution of all the samples in the data volume, which can more clearly show the whole view and details of the data field, with no need to construct the intermediate geometric primitives. Compared with methods of footprint evaluation[4], shear-warp [5] and frequency domain volume rendering[6], the ray casting algorithm can generate high quality images and is easy to be transplanted to GPU hardware to accelerate computation because of its high parallelism[7]. By applying GPU-accelerated volume rendering techniques to the visualization of three-dimensional seismic data, it will be more convenient to observe and analyze the rock conditions and the layer surfaces information underground, improving the accuracy of seismic interpretation and real-time interactive performance.

II. RAY CASTING ALGORITHM OF 3-D DATA VOLUME

A. definition of data volume

3-D seismic data set can be represented as a binary tuple like K= (U, R), where U means the object set (i.e. sampling point set) and R is property set. For $\forall r \in R$, there is $r: U \to V_r$, where V_r means range of property r. That is, for $\forall p \in U$, there exists $r(p) \in V_r$. For instance, if R={x,y,z,a},where x,y and z mean coordinate property of sample points and a means numerical property, then there exists a one-to-one mapping between point p and <x(p),y(p),z(p)>.Moreover, there is an equivalent relation between the maps $f: \{x,y,z\} \to V_a$ and $a: U \to V_a$, i.e. f(x(p),y(p),z(p))=a(p). Data set K=(U, {x,y,z,a}) is called 3-D data volume(i.e. data field).

B. Division of data volume

For visualization of the data field, data set has to be divided in order to demonstrate the distribution of the data by different colors. As to 3-D data volume K= (U, R),U can be divided into m subsets such as $U_1, U_2,\ldots,U_m$ according to the property set R, where the subsets need to meet conditions as follows:①for $\forall u,v \in U_i$， $w \in U_j$ (i,j=1,2,..,m, and i≠j)，there exists Sim(u,v)≤δ, Sim(u,w)>δ and Sim(v,w)>δ（here Sim is a similarity function and δ means a threshold of similarity）;② $U = \bigcup_{i=1}^{m} U_i$, $U_i \cap U_j = \phi$.

The division of U can be expressed as U/Sim={$U_1,U_2,\ldots,U_m$} in the visualization process of 3-D data volume. After setting some thresholds like d_i(i=0,1,…,m) that meet the partial order, Sim can be simplified as $|a(u)-a(v)| \le \delta$， where a(u)， $a(v) \in [d_i, d_{i+1}]$.

Palette set CP={$C_1,C_2,\ldots,C_m$} and opacity set NP={$\alpha_1, \alpha_2,\ldots, \alpha_m$} can be set based on the division U/Sim={$U_1,U_2,\ldots,U_m$},where cp: U/Sim→CP and np: U/Sim→NP are one-to-one mappings, i.e. $cp(U_i)=C_i$, $np(U_i)= \alpha_i$. Thus, for $\forall p \in U_i$, it has the color C_i and opacity α_i

C. Enhancement function of the layer surface

There are some surfaces constituted by the same or similar data in the data volume(i.e. layer surfaces in the data field). Variation and distribution features of layers in the data field can be showed more clearly by adjusting the transfer function of opacity(i.e. the layer surface enhancement function).According to the chosen layer value f_v and opacity α_v if $f(p)=f_v$, where $f(p)$ is the data value of sampling point p,

Published by Atlantis Press, Paris, France.

then set its opacity as α_v. For those sampling points whose data values near to f_v, the opacity is set close to α_v; otherwise, the opacity is set at zero. The formula for computing the opacity is:

$$\alpha(p)=\alpha_v * \begin{cases} 1 & f(p)=f_v \\ 1-\dfrac{k(p)*|f(p)-f_v|}{r} & |f(p)-f_v|\le r \\ 0 & others \end{cases} \quad (1)$$

In the formula, $k(p)$ is the gradient function associated with data value of the sampling point p, which evaluates degree of decrease of opacity$\alpha(p)$ with the data value away from f_v; r means the thickness of the layer surface to be displayed.

If r is too large, it will be difficult to highlight the variation of the layer surfaces; otherwise, if r is too small, there will be illusive fractures and cavity. The appropriate layer image can be displayed by adjusting r interactively when the data volume is actually rendering. And also, layer surfaces of different characteristics will be showed through choosing different f_v, $k(p)$ and r at the same time.

D. Color and opacity of resampling point

In the ray casting algorithm, a ray will emit from each screen pixel in the sight direction and a series of resampling points are chosen at each location along the ray [8]. The color and opacity of a resampling point can be calculated from its eight neighboring sampling points by tri-linear interpolation. Assume that there is a resampling point p in the cuboid whose eight vertexes are $p_i\ (i=1,2...8)$ [2](Figure 1). Besides,

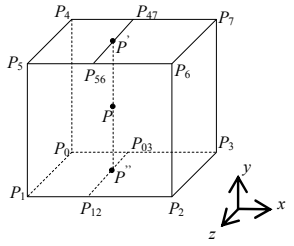

Figure 1. Tri-linear interpolation[2]

the projection point of p on the upper and lower planes of the cuboid is p' and p", whose projection point on the front and back planes is p_{56}、p_{47}、p_{12}、p_{03}, then g(p) ,which means the color or opacity of p, can be calculated as follows:

$$g(p)=\frac{y_{p'}-y_p}{y_{p'}-y_{p''}}g(p'')+\frac{y_p-y_{p''}}{y_{p'}-y_{p''}}g(p') \quad (2)$$

Similar to formula (2), g(p') and g(p") are calculated by projection points p_{56}、p_{47}、p_{12}、p_{03} along y-direction while $g(p_{ij})$ $(0 \le i, j \le 7)$ can be calculated by $g(p_i)$ and $g(p_j)$ along x-direction. At last, g(p) is figured out by g(p')and g(p"), after 7 times of linear interpolation.

E. Compositing color and opacity

In the ray casting algorithm, the accumulated opacity will increase as the colors and opacities of resampling points are composited in front-to-back order. When the accumulated opacity is larger than 1.0, the compositing can be terminated as the following voxel will not contribute to the color of the pixel. Assume that the color and opacity of the current resampling point are C_{now} and α_{now} and the input color and opacity are C_{in} and α_{in}, then the formula to calculate the output color C_{out} and opacity α_{out} is [7]:

$$\begin{cases} C_{out}\alpha_{out}=C_{in}\alpha_{in}+C_{now}\alpha_{now}(1-\alpha_{in}) \\ \alpha_{out}=\alpha_{in}+\alpha_{now}(1-\alpha_{in}) \end{cases} \quad (3)$$

F. Rendering pipeline of ray casting algorithm

Considering principles discussed above, the primary rendering pipeline of ray casting algorithm is:

(1) Make a division of the data volume K=(U, R) according to the similarity function Sim, denoted as U/Sim;

(2) Set palette set CP and opacity set NP. For $u \in U_i$, $U_i \in U/Sim$, u is given a color $cp(U_i)$ and opacity $np(U_i)$;

(3) A ray will be cast from each screen pixel in the sight direction and a series of resampling points denoted as {p} are chosen at each location along the ray. For each resampling point p, its color and opacity is calculated by tri-linear interpolation.

(4) Composite the colors and opacities of resampling points along the rays to get the colors and opacities of pixels, which form the image of visualization, showed in the screen.

III. VOLUME RENDERING BASED ON GPU

GPU consists of more processing cores. For ray casting algorithm based on GPU, a 3D texture will be created according to the colors and opacities of sampling points and the process of resampling and compositing will be handled by vertex and fragment shaders.

A. Resampling and compositing on GPU

For a ray which enters the data volume at p_{in} and exits at p_{out}(Figure 2), the position of the i-*th* resampling point denoted as p_{iNow} can be calculated by the formula as below:

$$p_{iNow}=p_{in}+i*d*normalize(p_{out}-p_{in}) \qquad i \in N \quad (4)$$

In the formula, d means step distance between resampling points and $normalized(p_{out}-p_{in})$ means the normalized vector of $(p_{out}-p_{in})$. If $|p_{iNow}-p_{in}|>|p_{out}-p_{in}|$, then the resampling point is outside of the data volume.

In order to get the intersection p_{in} and p_{out}, a bounding box of the same size as the data volume has to be rendered into the frame buffer before resampling [11] [12], whose eight vertexes are endowed with space coordinate v_i, texture coordinate vt_i and color c_i, which meet the requirement $v_i=vt_i=c_i$(i=1,2…8 and the values of v_i, vt_i and c_i are 0.0 or 1.0). When the rendering is over, for any point $A(v, vt, c)$ on the plane of the bounding box(v, vt and c are space coordinate, texture coordinate and color of $A(v, vt, c)$), there also exists the character like $v=vt=c$. Thus, the space and texture coordinate of a point on the plane of the bounding

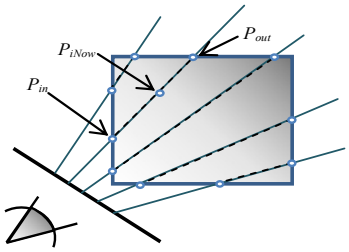

Figure 2. resampling of ray casting algorithm [11]

box can be obtained from the color buffer after rendering the bounding box. If the back faces were culled while rendering, the positions of entering points p_{in} are stored in the color buffer; otherwise the front faces were culled, there will be the exiting positions p_{out} to be stored.

The color and opacity of the i-*th* resampling point can be obtained by a lookup of the 3D texture according to the position p_{iNow}, calculated by formula (4) with the intersection p_{in} and p_{out} and the compositing will be processed on the fragment shader with the formula (3).

B. Pipeline of volume rendering on GPU

GPU accelerated volume rendering needs to transfer the intersections between rays and bounding box and the step distance d to the Cg shader program as parameters, whose primary rendering pipeline are(Figure 3):

(1) Create 3D texture according to the colors and opacities of sampling points after the division of the three-dimensional data field K=(U,R);

(2) Render the bounding box for the first time and cull the front faces to obtain the exiting intersections p_{out};

(3) Unlock the function of vertex and fragment shaders and cull the back faces while rendering the bounding box for the second time, meanwhile, the intersections p_{in} can be obtained from the texture coordinates;

(4) Resampling and compositing on the fragment shader. If the accumulated opacity is larger than 1.0 or the resampling point is outside of the data volume, terminate the resampling and compositing and return the accumulated color, otherwise, continue to compute the new resampling point after compositing;

(5) Refresh the buffer and show the visualized image to screen.

IV. DATA VISUALIZATION FOR SEISMIC INTERPRETATION

A. Implementation and result

The GPU accelerated ray casting algorithm for 3D seismic data visualization was implemented where the experimental environment are as follows:

System: win 7 X64;

Graphics card: NVIDIA GeForce GT 330M, 1G

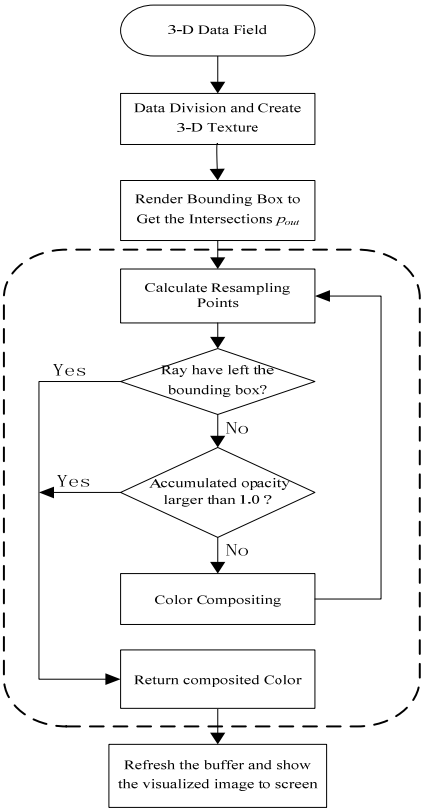

Figure 3. Pipeline of volume rendering on GPU

memory;

CPU: Intel Core i3 M350, 2.27 GHz;

Size of primary memory: 6G;

API: openGL including the extensive library glew and the shading language Cg.

The size of data volume in the following experiment is $101\times101\times500$ which was cut from a SEG-Y file and the data was divided into 256 levels.

(1) Visualization of gray mode: set colors of grey for every sampling point according to the division of the data volume. As it is showed in Figure 4, visualization of gray mode could roughly demonstrate distribution characteristics of the three-dimensional seismic data field.

(2) Visualization of pseudo-color mode: set gradual colors of red, yellow, green and blue for the sampling points where the seismic values of the red points are the largest and the yellow are smaller, while the blue are the smallest. It is obvious that the data distribution characteristics are more clearly to be observed in the visualized image of pseudo-color mode showed in Figure 5.

(3) Visualization of layer-highlight mode: According to formula (1), set $k(p)$=1.0, r=60.0, α_v=1.0 and f_v to be the maximum of seismic data field. The visualized image of layer-highlight mode is showed in figure 6,

where the features of the layer surfaces in the data field are more distinct and intelligible.

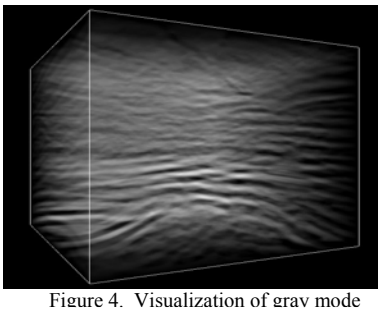

Figure 4. Visualization of gray mode

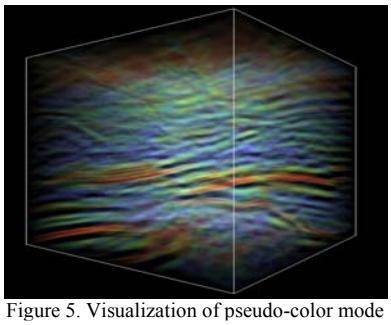

Figure 5. Visualization of pseudo-color mode

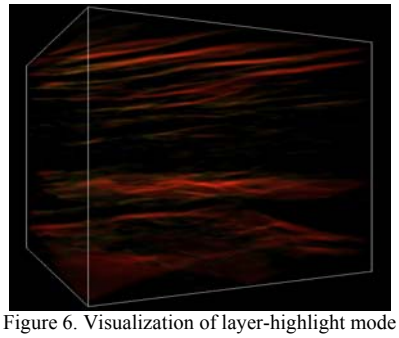

Figure 6. Visualization of layer-highlight mode

B. *Efficiency*

(1) Experiment at different viewpoints of a same data set

For examining the interaction response time of ray-casting algorithm based on both CPU and GPU, a seismic data volume of 101×101×250 was observed at ten different eye positions. In order to decrease the influence of the accidental factors in the operating environment, 50 trials were taken for each viewpoint to get the average running time after removing the maximum and minimum. The experiment result is shown in figure 7, where all the GPU running time is in the interval of [14.1886, 15.7046] and the average value is 14.9947(ms), very faster than the CPU running results of [9250.04, 9737.42] and 9486.24(ms), proving that the GPU accelerating algorithm has much stronger real-time efficiency. The main reason of this discrepancy is that the CPU algorithm has much bigger computational complexity for the intersection calculation between the ray and bounding box, the resampling and the compositing. However, For the GPU algorithm, the intersections p_{in} and p_{out} can be obtained just by rendering the bounding box twice while culling the front and back faces, and the running time can be much reduced by doing the resampling and composting on shader. At different viewpoints, the positions and number of resampling points are different for the different intersections, and so the curve of response time has fluctuation features because of the discrepant computational complexity of resampling and compositing.

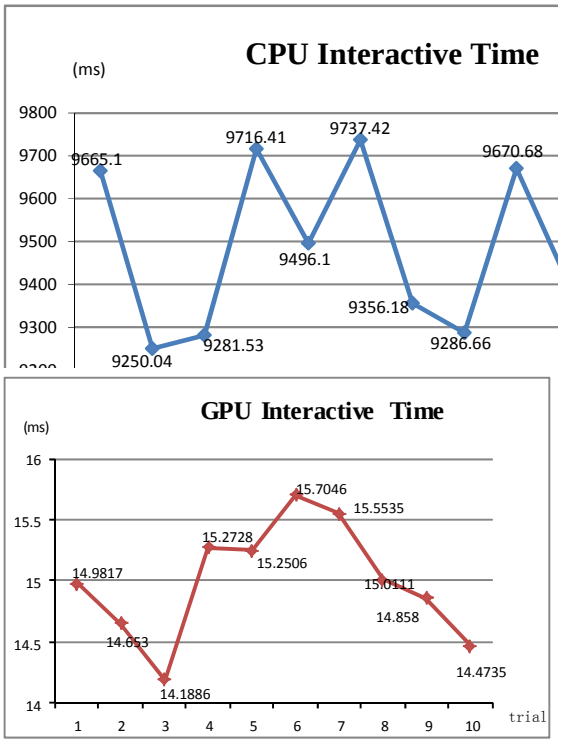

Figure 7. The contrast of CPU and GPU

(2) Experiment at the same viewpoint of different data sets

In this experiment, the data set was gradually increased ten times by an increment of 10×28×25 and observed at the same viewpoint. Just like the experiment above, for every trial, the average running time was calculated after removing the maximum and minimum in order to avoid the influence of the accidental factors and the result is shown in Figure 8. It can be seen that obviously the processing time of GPU algorithm is faster than CPU (the reason has been discussed above), and besides, what is not so obviously is that both the GPU and CPU algorithm processing time are slowly growing as the data set was increased every time. This can be explained that although the sampling points in the data set increased for every trial, the intersections between the ray and bounding box and the number of the resampling points in the data volume didn't grow, and so the computational complexities of the tri-linear interpolation and the compositing are approximately the same. This proves that the increase of the data set has little impact on the processing time of the ray casting algorithm of the same viewpoint. However, as the increasingly occupied memory of the RAM and graphics card may need more addressing time, and for every trial, the computing resource acquired is different, the

Published by Atlantis Press, Paris, France.

curves of the GPU and CPU processing time show slowly wavelike rising trend.

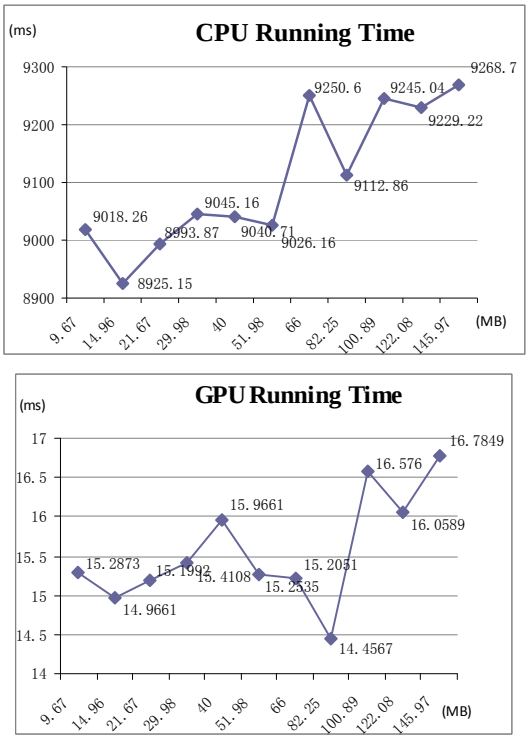

Figure 8. The contrast of CPU and GPU ray casting algorithm running time as data set increase

V. CONCLUSIONS

Principles of volume rendering and methods of GPU accelerating are outlined based on the formal definition and division of the data volume. Data visualization for seismic interpretation of grey and pseudo-color mode has overcome the one-sidedness and deficiency of profile, slice and surface rendering methods for it reflecting the overall distribution of the data in the data field. And also, the features of the layer surfaces can be highlighted through the design of transfer function. Because of the powerful floating computing ability of GPU, the speed to form an image by the ray casting algorithm is greatly improved. Thus, it is very convenient to adjust the transfer function and the eye point in real-time to observe different parts of the data field, meeting the real-time requirement of interactive.

ACKNOWLEDGMENT

We are greatly indebted to colleagues at College of Information Engineering, China University of Petroleum. We thank A/Prof. LU You for his useful suggestions and many interesting discussions.

This work is partly supported by the National High-Tech Research and Development Program under Grant No. 2009AA062802, the Nature Science Foundation of China under Grant No. 60473125, the Innovation Foundation of CNPC under Grant No. 05E7013; National Key Project Foundation of Science under Grant No. G5800-08-ZS-WX.

REFERENCES

[1] Gerald D. Kidd. “Fundamentals of 3-D seismic volume visualization,” The Leading Edge, vol.18, No.6,1999, pp: 702-709.

[2] Zhang Erhua, Gao Lin, Ma Renan, Yang Jingyu. “Principles and Implementation of Visualization on 3D Seismic Data Volume,” CT Theory and Applications, vol.16, No.3, 2007, pp:20-28.

[3] Ming Jun, Peng Gang , Shen Zhanghong, Liu Chuncheng. “All 3-D Interpretation Method and Its Application in BZ25-1 Oil Field,” China Offshore Oil and Gas (Geology),vol.15, No.6,2001, pp:416-422.

[4] L.Westover. “Footprint evaluation for volume rendering,” Computer Graphics,vol.24, No.4, 1990, pp:367-376.

[5] P.Lacroute, M.Levoy. “Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation,” Proc. SIGGRAPH '94, Jul.1994, pp:451-458.

[6] T.Totsuka, M.Levoy. “Frequency Domain Volume Rendering,” Proc. of SIGGRAPH '93, Aug.1993, pp:271-278.

[7] Tang Zesheng, et al. Visualization of 3D data field 1rd ed .Beijing: China Machine Press, 1999.

[8] M.Levoy. “Display of surfaces from volume data,” IEEE Computer Graphics and Applications, Vol.8, No.3, 1988, pp:29-37.

[9] P.Sabella. “A rendering algorithm for visualizing 3D scalar fields,” Computer Graphics, Vol.22, No.4, 1988, pp:51-58.

[10] M.Levoy. “Efficient Ray Tracing of Volume Data,”ACM Transactions on Graphics, Vol.9, No.3, 1990, pp:245-261.

[11] Markus Hadwiger, Patric Ljung, Christof Rezk Salama, Timo Ropinski. “Advanced Illumination Techniques for GPU-Based Volume Raycasting,” Proc. SIGGRAPH Asia '08, 2008.

[12] J. Krüger , R. Westermann. “Acceleration Techniques for GPU-based Volume Rendering,” Proc.14th IEEE Visualization Conference (VIS’03), 2003, pp:287-292.