Dening LUO, Yi LIN, Jianwei ZHANG, 2021. GPU-based multi-slice per pass algorithm in interactive volume illumination rendering. *Frontiers of Information Technology & Electronic Engineering*, 22(8):1092-1103.
https://doi.org/10.1631/FITEE.2000214

GPU-based multi-slice per pass algorithm in interactive volume illumination rendering

Key words: Volume rendering; Volume illumination; Volumetric datasets; Multi-slice per pass

Corresponding author: Dening LUO
E-mail: onexinoneyi@hotmail.com
ORCID: https://orcid.org/0000-0003-4359-5975

Motivation

1. To obtain an improved three-dimensional (3D) shape perception of volumetric datasets, a better realistic volume illumination algorithm needs to be considerably studied.

2. The calculation overhead associated with interactive volume rendering is unusually high, and the solvability of the problem is adversely affected when the data size and algorithm complexity are increased.

3. Currently, understanding the manner in which the GPU and parallel rendering techniques can be used to resolve performance issues has become a significant challenge.

Main idea

1. The algorithm dynamically renders an arbitrary number of slices in a single pass of GPU to improve the calculation efficiency by avoiding frequent draw calls.

2. The algorithm is based on the traditional graphics pipeline and can be easily combined with other polygonal algorithms, thereby offering considerable interactivity without affecting the rendering quality.

3. The algorithm can effectively produce global volume shadow and translucent effects of volume illumination for the real-world volumetric datasets, obtaining a better perception of the shape and depth of volumetric datasets.

Method

1. Volume rendering is usually designed on the basis of the whole rendering framework, which includes the process of volumetric datasets, specific algorithm flow, and data updates.

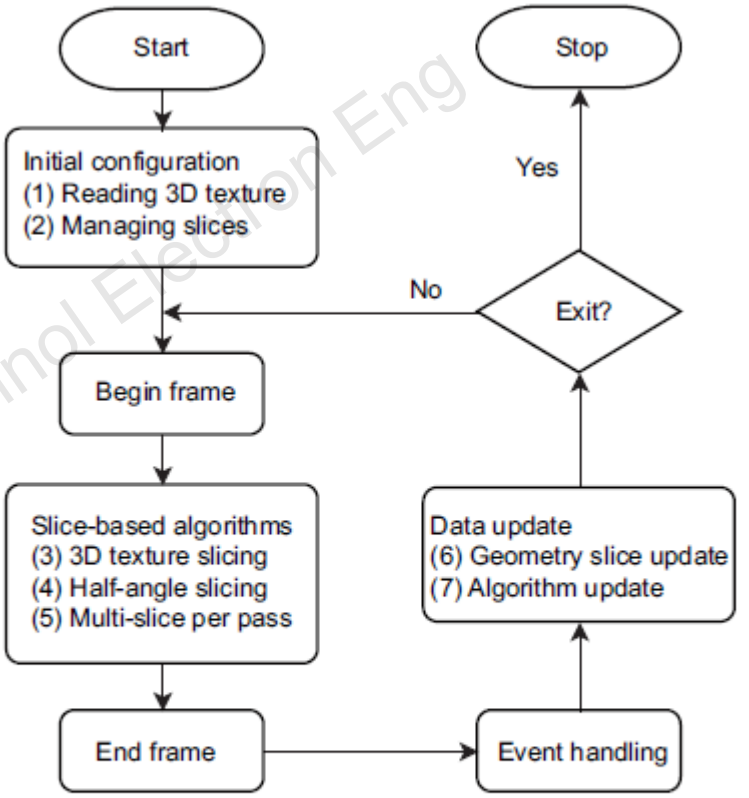

Fig. 1 The multi-slice per pass (MSPP) rendering framework. In the initial configuration, the slice-based algorithms and data updates are involved in the graphics pipeline

Method (Cont'd)

2. MSPP makes sure that multiple slices could be rendered in one pass to reduce alternate calculating and rendering on CPU and GPU.

Algorithm 2 MSPP for volume shadow on GPU

```
Initialize n, LightBuffer_{0,1,...,n-1}(1,1,1,1),
EyeBuffer(0,0,0,0), and AttenuationBuffer(1,1,1,1)
for int i=0; i <TotalPasses; i++ do
    // Step 1: render n slices per pass into
    // LightBuffer_{0,1,...,n-1}
    Bind LightBuffer_{0,1,...,n-1} as the render target
    for each sample do
        for each slice per pass do
            Evaluate sample color α
            Multiply α by illumination color and output
            to the corresponding light buffer
        end for
    end for
    // Step 2: render n slices per pass and accumulate
    // into EyeBuffer
    Bind EyeBuffer as the render target
    Bind LightBuffer_{0,1,...,n-1} as texture
    Bind AttenuationBuffer as texture
    for each sample do
        for each slice per pass do
            Compute LightBuffer_{0,1,...,n-1} texture
            coordinates
            Evaluate sample color
            Read illumination intensity from
            LightBuffer_{0,1,...,n-1} and blend with illumi-
            nation attenuation of the previous slices
            Multiply the color by illumination intensity
        end for
        Blend all slices and accumulate into EyeBuffer
    end for
    // Step 3: accumulate illumination attenuation of
    // all rendered slices
    Accumulate illumination attenuation slice by slice
    into AttenuationBuffer
end for
```

Method (Cont'd)

3. In the four components (x, y, z, w) of each vertex, the fourth component w can be used to mark the slice number. Unfortunately, the vertex will be normalized in the rendering pipeline, so the first three components need to be multiplied by the slice number to ensure correctness in the shader. Meanwhile, the coordinates of the vertices in the shader need to be divided by the component w to restore the correct position.

Method (Cont'd)

4. Another question is that each slice of MSPP outputs correct light attenuation messages into LightBuffer$_{0, 1, \ldots, n-1}$. First, one slice per pass is determined by the slice number mark (MarkNo) of the vertices and the current pass (PassNo). Second, once a slice is calculated in the fragment shader, the current slice is computed and outputted to the corresponding buffer:

$$|\mathrm{MarkNo} - n \cdot \mathrm{PassNo} - i| < \mathrm{Threshold}, \qquad (7)$$

where "Threshold" is the threshold value and *i* is one slice of *n* slices.

Major results

Performance comparison

Table 1 Performance comparison of the three methods without shadow in different numbers of slices for Engine data

Case	Average time spent (ms)					
	16	32	64	128	256	512
A	1.00	1.04	1.05	1.06	1.15	1.61
B	1.25	1.77	2.78	4.81	9.58	19.34
C	1.05	1.06	1.34	1.85	2.87	5.28
D	1.06	1.06	1.10	1.33	1.85	2.93
E	1.08	1.07	1.15	1.33	1.69	2.30

A: view-aligned slicing; B: half-angle slicing; C: MSPP with 4 slices per pass; D: MSPP with 8 slices per pass; E: MSPP with 16 slices per pass. The data are divided into six different numbers of slices from 16 to 512 increased by a power of 2

Table 2 Performance comparison of the two methods with volume shadow in different numbers of slices for Engine data

Case	Average time spent (ms)				
	32	64	128	256	512
A	2.84	4.69	8.93	19.13	41.70
B	1.87	3.04	5.25	10.24	19.52
C	1.54	2.29	3.45	7.01	12.22
D	1.47	2.08	3.34	5.81	10.62
E	1.62	2.42	3.91	6.95	14.71
F	1.80	2.72	4.41	7.77	13.86

A: half-angle slicing; B: MSPP with 2 slices per pass; C: MSPP with 3 slices per pass; D: MSPP with 4 slices per pass; E: MSPP with 5 slices per pass; F: MSPP with 6 slices per pass. The volume data are divided into five different numbers of slices from 32 to 512 increased by a power of 2

Major results (Cont'd)

Rendering effects

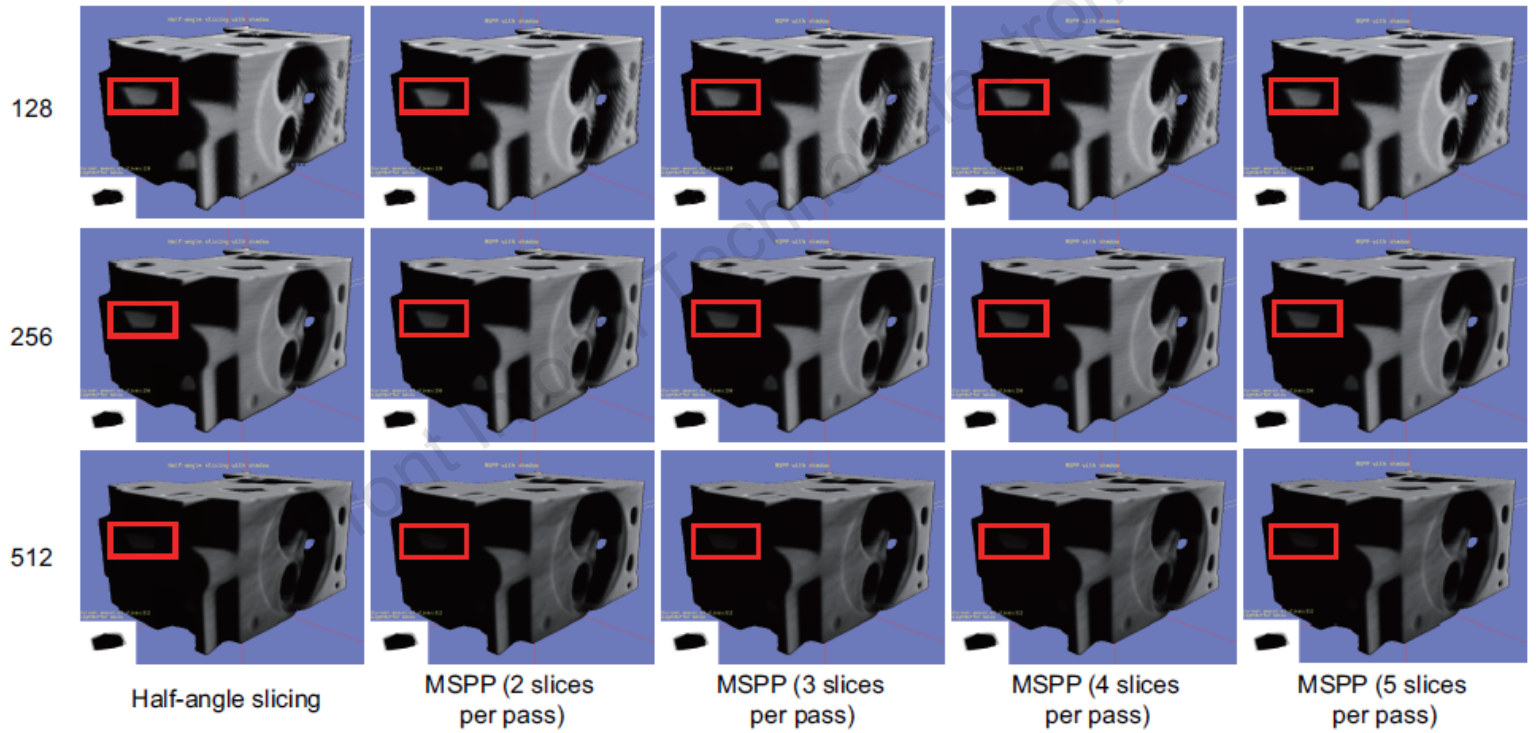

Fig. 7 Effects of volume shadow of half-angle slicing and MSPP with different numbers of slices per pass. The illuminated area within the red frame is the case where the light source is calculated through the space structure of volumetric datasets. Meanwhile, images at the bottom left are the illumination attenuation buffers of the last slice

Major results (Cont'd)

Rendering effects

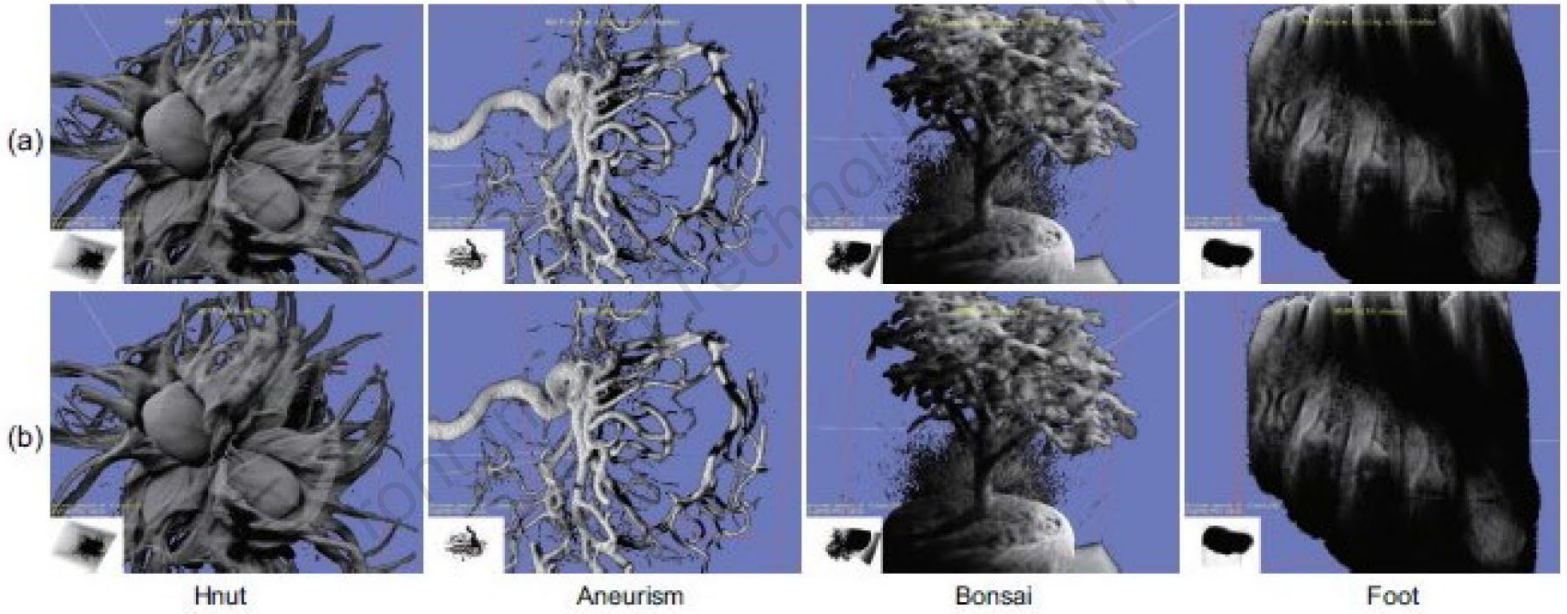

Fig. 9 Effect comparison for volume shadow between half-angle slicing (a) and MSPP (b). The four volume datasets from the left to right are Hnut (resolution: 512^3; file size: 128 MB), Aneurism (resolution: 256^3; file size: 16 MB), Bonsai (resolution: 256^3; file size: 16 MB), and Foot (resolution: 256^3; file size: 16 MB). Meanwhile, images at the bottom left are the illumination attenuation buffers of the last slice

Conclusions

1. The MSPP method of slice-based volume rendering can quickly implement the visualization of volumetric datasets and some volume effects, such as translucent effect with TF and volume shadow.

2. In contrast to other slice-based volume rendering algorithms such as half-angle slicing, MSPP can improve the rendering performance for volume shadow by at least two times and effectively reduce the memory overhead. Meanwhile, the scalability of MSPP can be flexibly applied to more volumetric datasets and rendering applications.

3. Combined with the parallel rendering framework, it also dramatically improves the performance of volume rendering.

Dening LUO is a PhD candidate at Sichuan University and a visiting student at Zurich University in 2019. He received the BS degree in Computer Application from Nanchang University in 2011 and the MS degree in Computer Science from Sichuan University in 2014. His research fields are real-time computer graphics, volume rendering, and parallel rendering.

Jianwei ZHANG received the BS degree in 1993, the MS degree in 2000, and the PhD degree in 2008, in Computer Application from Sichuan University, China. He is a PhD tutor. His research interests are computer graphics, virtual reality, machine vision, and air management visualization. He has published over 30 papers at home and abroad.