Accelerate Volume Splatting by Using Run Length Encoding

Zhang Jiawan, Sun Jizhou, and Sun Zhigang

IBM Computer Technology Center, School of Electronic and Information Engineering, Tianjin University, 300072 Tianjin, P. R. China
{jwzhang, jzsun, zhgsun}@tju.edu.cn
http://jwzhang.yeah.net

Abstract. Methods such as splat hierarchies, indexing and lists have been presented by the research society in recently years, to accelerate the splatting, a popular volume rendering algorithm. In this paper, a run length encoding (RLE) accelerated, pre-classification and pre-shade sheet buffer volume splatting algorithm is presented, which can enhance the speed of splatting without trading off image quality. This new technique saves rendering time by employing RLE mechanism so that only voxels of interest are processed in splatting. RLE based data structures are defined to exploit spatial coherence of volume and intermediate rendering images. A fast and accurate sheet buffer splatting method is used in the rendering process, which accelerates the splatting by traversing both the voxel scanline and the image scanline in sheet buffer simultaneously. Experiments practice proves that RLE can efficiently skip over transparent voxels in splatting and high speedup can be obtained by using the proposed algorithm.

1 Introduction

Volume Rendering [2] has been a new branch of computer graphics and an important visualization technique in recent years. It has gained great popularity in the comprehension and visualization of volumetric data sets from medical imaging devices such as computer tomography (CT), magnetic resonance imaging (MRI), ultrasound, positron emission tomography (PET), single photon emission tomography (SPET) and scientific simulations such as computer fluid dynamic (CFD).

A number of volume rendering algorithms such as volume ray casting [3], splatting [5] [6], shear-warp [7], and frequency domain methods [8] have been presented by the research society in the past two decades. Splatting proposed by Lee Westover [5] is an object-order traversal algorithm. In this algorithm, the voxels are sorted slice by slice in the front-to-back or back-to-front order. Traversed in the order, each voxel is classified by using a proper transfer function, and shaded by a given illumination model. Then, the voxel is projected into the image plane, and its contribution is accumulated to an image buffer using a projected reconstruction kernel called footprint. In this way, successive slices are composited to produce the final image.

Generally, there are two important software optimizations to accelerate volume rendering algorithm. One is early ray termination, which is most easily implemented in a ray caster. In this method, the algorithm traces each ray in front-to-back order and terminates when the accumulated ray opacity reaches a threshold close to full opacity. Any additional voxels reached by the ray are occluded, so they need not be rendered. The other is coherence acceleration using spatial data structures in which transparent voxels are skipped and only non-transparent voxels are considered in contribution to the final image. Because typical classified volumes have only 5%-30% percentage of non-transparent voxles [3] , and the computational cost is linearly proportional to the size of the voxels rendered in splatting algorithm, high speedup can be obtained by using an efficient spatial data structure.

Several algorithms are proposed to speed up volume splatting. Laur and Hanrahan [9] use a pyramidal volume presentation to improve the speed of splatting. Given transfer function and view-independent shading functions, an octree is constructed in which each node contains the average RGBA value of all its children and a value indicating the average error associated with the average. Then, the octree is traversed in a viewing order to splat the voxels, depending on the given allowable error that determines the refinement of rendered images. This algorithm does reduce rendering costs, but trades off image quality. Ihm [10] demonstrated a simple but efficient technique to speed up splatting by using indexing techniques. In this method, a sequence of pointers corresponding to a particular density value in one slice is enumerated in the increasing order of density values. The pointer associated with one density is a data block that keeps the position offsets which have the density value. In the rendering process, one can easily visit only voxels with values of interest by binary search of density values that are between required ranges of density values. This method can efficiently exploit spatial coherence and gain high speedup. One trivial disadvantage is that the algorithm requires much extra memory.

RLE based data structures used in Lacroute's shear-warp algorithm [7] has been proved a successful data structure, which accelerates the rendering process by efficiently skipping over transparent voxels. However, RLE based method is traditionally difficult to be applied in object-order volume rendering algorithms. In this paper, we propose a RLE accelerated, pre-classification and pre-shade sheet buffer based volume splatting algorithm. By employing RLE based data structures, together with a sheet buffer based splatting rendering process, the new algorithm can obtain high speedup without trading off image quality.

2 Basic Algorithm

Our algorithm consists of four steps, as shown in Fig. 1, pre-classification of raw density volume, pre-shade, run length encoding of classified volume, and sheet-buffer based splatting. In the following subsections, we will explain each step in detail.

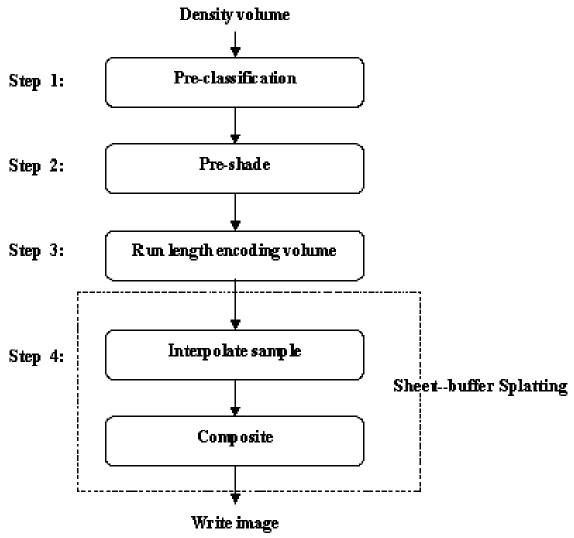


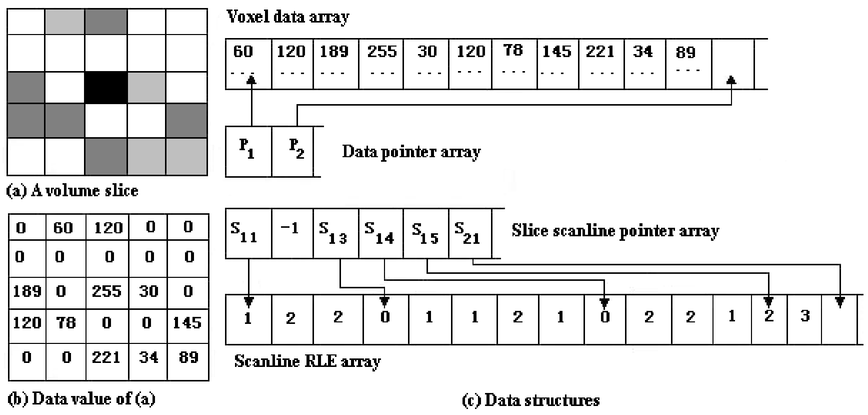
Fig. 1. Basic algorithm

2.1 Pre-classification and Pre-shade

Step 1 and step 2 of Fig.1 are carried out once the raw density volume is loaded. In the pre-classification process, the transfer function is used to map the density value to proper opacity. In the pre-shade process, illumination model such as Phong [1] is used to calculate RGB color channel from the view parameters and local gradient approximated normal. Here, we adopt pre-classification and pre-shade scheme, because both two steps can be done in the pre-processing step, which can save much rendering time. The number of visible voxels that require pre-shading is less than the number of pixels that require post-shading. Furthermore, the number of voxels to be projected in pre-shading splatting is lower than that of post-shading, too.

2.2 RLE Data Structure for Spatial Coherence

Run length encoding is used to encode the volume in the new algorithm. The RLE data structures created are shown in Fig. 2. Here, four tables are defined: a slice scanline pointer array, a data pointer array, a scanline RLE array and a table stores data of all the non-transparent voxels. The main difference compared to Lacroute's method is that in our algorithm we separate the slice pointer array with data pointer array and RLE is used in the scanline level.



0	60	120	0	0
0	0	0	0	0
189	0	255	30	0
120	78	0	0	145
0	0	221	34	89

Fig. 2. Example data structures by using run length encoding. (a) A volume slice example (b) Data of example in (a), values indicate the grey scales of pixels. (c) RLE based data structures created in our algorithm.

The scanline RLE array contains sequences of transparent run and a non-transparent run that is contructed in the slice-by-slice, scanline-by-scanline order. Each entry in the slice scanline pointer array points to the first transparent run of that scanline, for example, Sij points to first transparent run length of jth scanline in ith slice. If all voxels of a scanline are transparent, the flag value -1 will be assigned as the pointer. Each pointer in the data pointer array is responsible for all non-transparent voxels in a slice. Here, because transparent voxels do not contribute to the final image, their data values are not stored in the voxel data array any more to save memory. Each entry of the voxel data array contains opacity and color information of a non-transparent voxel. Since we adopt a pre-classification and pre-shade scheme, these optical properties can be pre-computed during the pre-processing steps.

In addition, to make the splatting process view-independent, three sets of such data structure are built in the pre-processing step, one for each principal volume face along sheet-buffer, as shown in Fig. 3.

The new design of data structures has advantages compared to conventional method. First, the two pointer arrays in our method ensure flexible random access to any voxel scanline and slice. Second, the separation of run lengths and voxel data avoids alignment problems in some particular processor architectures. At last, low memory cost is maintained since RLE efficiently skips transparent voxels. Furthermore, since the traversal order of voxels is basically the same as the traditional splatting algorithm, hence any improvement techniques, such as early opacity termination, can be easily applied.

2.3 Sheet Buffer Splatting Rendering

In this subsection, a RLE accelerated, sheet buffer splatting rendering algorithm is presented. Since the sheet buffer is constructed parallel to the principal axes, it is also parallel to the voxel scanline. Naturally we can accelerate the splatting by traversing both the voxel scanline and the image scanline in sheet buffer simultaneously. In our work, we use the voxel scanline RLE to achieve fast and accurate splatting rendering. Fig. 4 depicts our rendering method. Here, we only show one sheet buffer scanline. In practice, there maybe several scanlines affected by one voxel scanline, based on the selected 3D reconstruction kernel and corresponding footprint of splatting. Each non-transparent voxel is projected onto the sheet buffer and spreads its contribution to neighbor pixels, whereas transparent voxels are efficiently skipped without any operation. Care must be taken in the processing of overlap areas. In conventional splatting, over and under operation [11] is used since pixel overlapped by two voxels is composited during the processing, which always leads to artifacts. In our implementation, we adopt adding operation in the processing of overlapped pixels, which results in more accurate rendering.

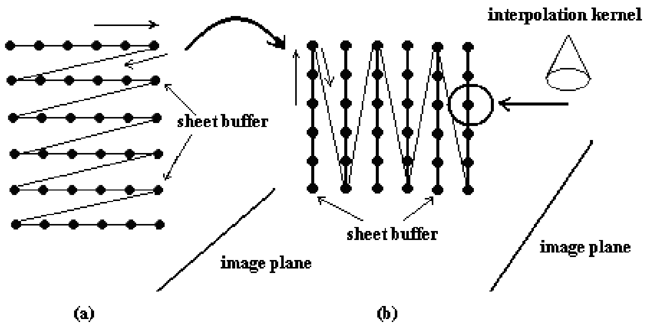


Fig. 3. RLE data structures created based on corresponding principal volume axes along sheet buffer. When the angle between the image plane and the principal volume axis is above 45^o, another set of data structures is created based on the other corresponding principal axis.

3 Experiment Results

We implement the proposed algorithm on a PC with 1 GHZ AMD Duron CPU, 256MB RAM. In our experiments, four CT, one X-ray,and one MRI volume data sets are used. Each pixel of these data sets is made up of 1 byte of gray tone.

Table 1 depicts the memory cost of RLE based data structures in our RLE based splatting, based on a given transfer function. The memory is about 3-5 times of the original volume storage. The reason is that we keep three copies of RLE based data structures in memory.

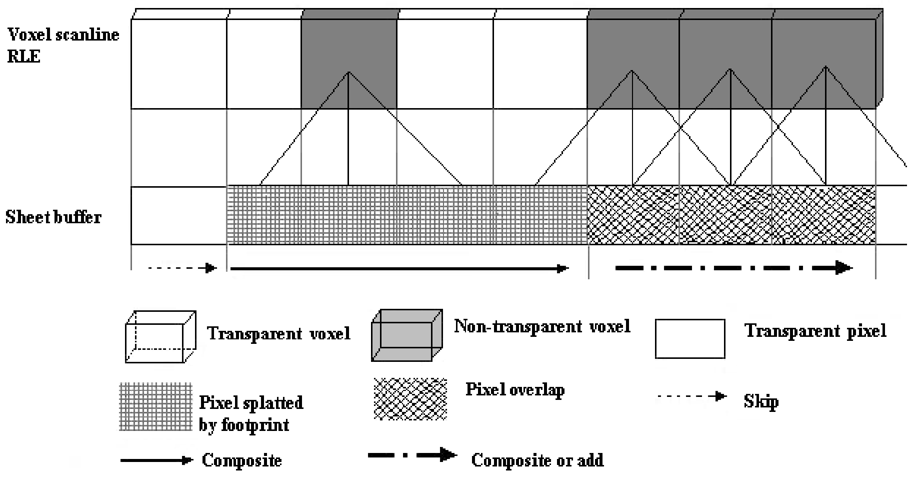


Fig. 4. Voxel scanline RLE based sheet buffer splatting rendering

Table 1. The memory cost of RLE based data structures

Data sets	Resolution	Volume size	RLE memory cost
CT head	128x128x128	1.9MB	5.1MB
CT lobster	320x320x34	3.32MB	12.4MB
MRI head	256x256x109	6.8MB	38.7MB
CT engine	256x256x110	6.9MB	39.1MB
X-ray foot	256x256x256	16.0MB	55.2MB
CT bonsia	256x256x256	16.0MB	60.1MB

Table 2. The speed comparison of splatting with and without optimization

Data sets	Non-RLE time	RLE time	Speedup
CT head	7.38s	0.41s	18x
CT lobster	12.21s	1.13s	10.8x
MRI head	22.48s	2.3s	9.8x
CT engine	24.31s	2.67s	9.1x
X-ray foot	53.88s	2.03s	26.5x
CT bonsai	55.04s	2.43s	22.7x

Table 2 summarizes the execution times of splatting with and without optimization. Experiments results demonstrates that high speedup can be obtained by applying RLE techniques into splatting even if at least three copies RLE data structures in memory. The rendering results in our experiments are shown in Fig. 5.

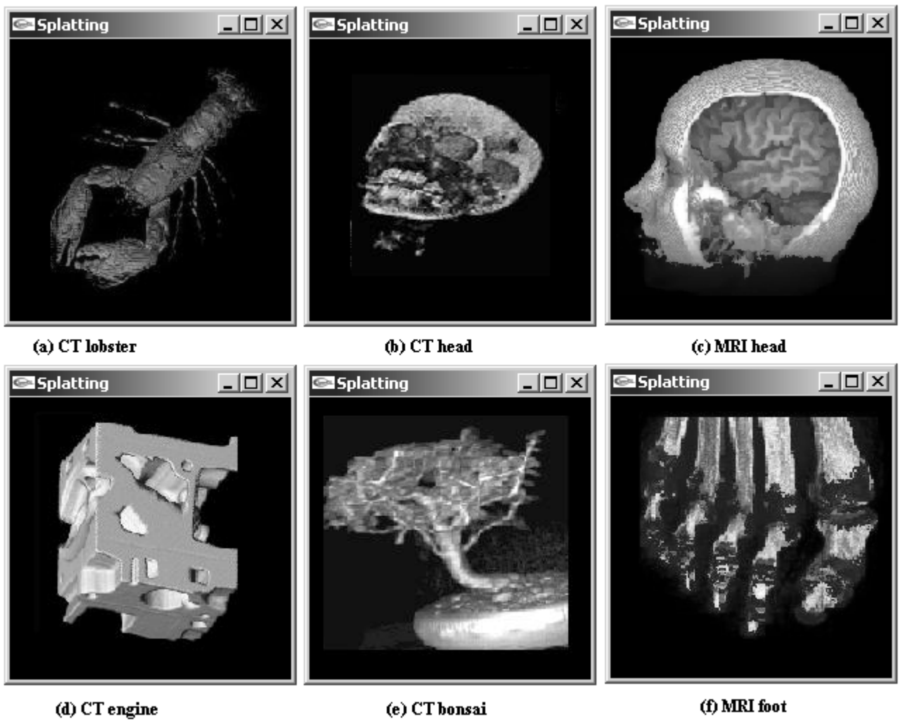


Fig. 5. Rendering results.(a) CT lobster, data set courtesy of Advanced Visual Systems, via Mark Kessler, University of Michigan Medical School.(b) CT head (c) MRI head,data set courtesy of Siemens Medical Systems, Inc., Iselin, NJ. (d) CT engine, data set courtesy of SoftLab Software Systems Laboratory, Department of Computer Science, University of North Carolina, Chapel Hill, NC.(e) CT bonsai,data set courtesy of S. Roettger, VIS, University of Stuttgart.(f) X-ray foot, data set courtesy of Philips Research, Hamburg, Germany.

4 Conclusions

We have presented a pure software accelerating method in the context of volume splatting algorithm. The method is based on applying RLE technique into the encoding of volume in order to exploit spatial coherence. Experiments practice proves that RLE can efficiently skip over transparent voxels. By defining RLE based data structures, together with a fast and accurate sheet buffer splatting rendering, high speedup can be obtained.

References

1. Phong B.T.: Illumination for Computer Generated Images. Ph.D disseration, University of Utah 1973
2. Drebin R. A., Carpenter L., Hanrahan P. : Volume rendering. Computer Graphics. **22** (1988) 65–74
3. Levoy M.: Efficient Ray Tracing of Volume Data. ACM Transactions on Graphics.**9**(1990)245–261
4. Kajiya J. T., Von Herzen B. P. : Ray Tracing Volume Densities. Computer Graphics. **18** (1984) 165–174
5. Westover L.: Interactive Volume Rendering. Volume Visualization Proc., Department of Computer Science, Univ of North Carolina, Chapel Hill.(1989) 9–16
6. Westover L. : Footprint Evaluation for Volume Rendering. Computer Graphics. **24** (1990) 367–376
7. Lacroute P., Levoy M.: Fast Volume Rendering Using a Shear-Warp Factorisation of the Viewing Transform. Computer Graphics. **8** (1994) 451–459
8. Totsuka T., Levoy M.: Frequency Domain Volume Rendering. SIGGRAPH '93, Anaheim, California.(1993) 271–278
9. Laur D., Hanrahan P.: Hierarchical Splatting: a Progressive Refinement Algorithm for Volume Rendering. Computer Graphics. **25** (1991) 285–288
10. Insung I., Kyoug R. L.: On enhancing the speed of splatting with indexing. IEEE Visualization' 95 Proc., Atlanta, Georgia.(1995)69–76
11. Porter. T., Duff T.: Compositing digital images. Computer Graphics, **18** (1984) 253–259