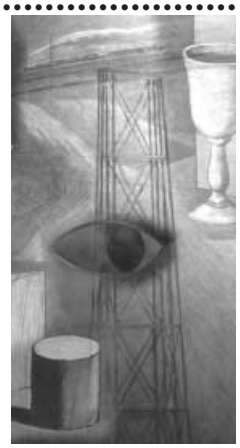
THE JOURNAL OF VISUALIZATION AND COMPUTER ANIMATION
J. Visual. Comput. Animat. **10,** 123–131 (1999)

A Selective Rendering Method for Data Visualization



By Wencheng Wang, Enhua Wu and Nelson Max*

Selective visualization is a solution for visualizing data of large size and dimensionality. In this paper a new method is proposed for effectively rendering certain chosen parts among the full set of data in terms of a colour buffer, referred to as the virtual plane, for storing intermediate results. By this method, scientists may concentrate their attention on the contents of data in which they are interested. Besides, the method could be easily integrated with all the current direct volume rendering techniques, especially progressive refinement methods and selective methods.

Received 20 May 1998; Revised 22 March 1999

KEY WORDS: refinement; volume rendering; scientific visualization; interaction

Introduction

The aim of scientific visualization is to give scientists a great insight of scientific data by mapping the data into intuitive images. In general, two main approaches are employed: extraction of isosurfaces[1] and direct volume rendering. Extraction of isosurfaces may produce clear-cut delineation of features, but the amount of information presented in the image is restricted. Direct volume rendering may take advantage of the nature of semitransparency in visualizing more volume contents, and in this method all the voxels in the volume are capable of contributing to the image. The methods for direct volume rendering can be classified into two categories: ray casting[2–8] and projection.[9–15] In the former method, colours of pixels are calculated by shooting rays through the data volume, and in the latter, cells are projected directly on to an image plane and the colour at each pixel is obtained by accumulating the colour effect of samples along a projection path.

So far a wide variety of techniques have been developed to support data exploration. However, as data are rapidly growing in size and dimensionality, most present techniques tend to become less usable, either because too many data are involved in the display or because an extensive search of a large data set is necessary to find interesting regions. A conceptual solution to this problem seems quite simple: to reduce the amount of data to be displayed or to increase the information content of the visualization. In this regard, various refinement and selective visualization methods have been proposed. For instance, an octree structure has been utilized to progressively render volumes,[16–18] and methods for selectively rendering particular features have been adopted for visualizing the interesting parts of data.[13,19–21] These methods are very useful as they help scientists watch their interesting contents quickly without wasting much time on the parts of no interest. However, they generally require a pre-identification of data features before rendering, and re-rendering from scratch is in general required each time the interesting part of data changes. This hinders scientists from understanding the data quickly, as scientists seldom know where the interesting part is within an unfamiliar data volume and therefore much time has to be spent on waiting for the images to be completely rendered.

In this paper we present a new selective visualization method. By 'selective' we mean that a selection criterion is applied to filter out interesting parts of the data for rendering. Here our emphasis is on the rendering speed of the selective part rather than on the selection creation discussed in many publications. Previous experience[16] has shown that most (around three-quarters) of the calculation in volume rendering lies in the work of sample interpolation and colour mapping. Aimed at efficiency promotion in this respect, a 2D colour buffer, referred to as the *virtual plane*,[22] is proposed in our method to store the intermediate results during the process of producing a

*Correspondence to: Wencheng Wang, Laboratory of Computer Science, Institute of Software, Academia Sinica, PO Box 8718, Beijing 100080, China.

Contract/grant sponsor: Natural Science Foundation of China.

 CCC 1049–8907/99/030123–09$17.50

series of images. By using the virtual plane, much of the work on sample interpolation and colour mapping can be saved. Besides, the intermediate results produced in the process could help scientists search for the interesting parts.

It is well known that rendering a volume from various positions and directions is very useful, and many interactive methods have been proposed[23–25] in this respect. However, as direct volume rendering is computationally intensive, measures for reducing the number of voxels involved in the computation have to be taken in these methods for interactive operations. Obviously, such measures will impede the comprehensive understanding of the data as the amount of information presented is correspondingly reduced. The method proposed in this paper provides a new approach that is able to render what is wanted among the full set of data. This is accomplished by adjusting the transparency values of voxels to highlight interesting parts for detailed analysis.

In the following section the formation and functions of the virtual plane are introduced. Then we discuss three tools working on the virtual plane for scientists to render interesting parts. The integration of the method with ray-casting and projection approaches is next described. Finally, experimental results and conclusions are given.

Virtual Plane

The *virtual plane* was first proposed by the present authors in Reference 22. It provides a new approach for colour composition, the fundamental operation in direct volume rendering. The method includes two steps. First, colours of voxels are distributed over a 2D plane, referred to as the *virtual plane*, along the viewing direction from front to back. Then an image is produced by an operation on the virtual plane using a compression frame. It works in detail as described in the following procedure.

As we know, the method to composite colours at each pixel is based on the principle of *over* operation as presented in Reference 26. The operation runs as described below.

Suppose a medium A overlaps a medium B along the viewing direction. Then the colour composited is calculated through the formula

$$C = C_a{*}O_a + (1 - O_a){*}O_b{*}C_b \tag{1}$$

Where C_a, O_a and C_b, O_b are the colour and opacity values for the media A and B respectively.

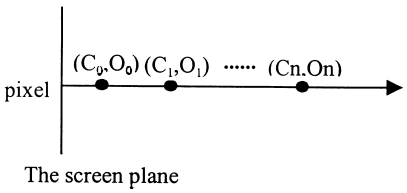


Figure 1.

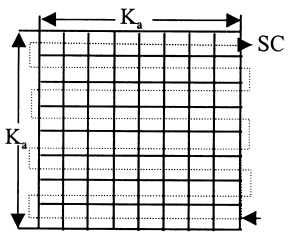


Figure 2.

Through a front-to-back traversal the recurrent formulae for compositing colours at each pixel can be expressed as[15]

$$C_{acc} = (1 - O_{acc}){*}C_{new}{*}O_{new} + C_{acc} \tag{2}$$

$$O_{acc} = (1 - O_{acc}){*}O_{new} + O_{acc} \tag{3}$$

where C_{new} and O_{new} are the colour and opacity of the newly calculated sample at a particular pixel and C_{acc} and O_{acc} are the accumulated colour and opacity of the pixel along the projection path.

By recurrently using equations (2) and (3) n times, we get the expression[2]

$$C = \sum_{j=1}^{n} C_j O_j \prod_{i=0}^{j-1} (1 - O_i) \tag{4}$$

where C_j and O_j are the colour and opacity of the jth sample along the accumulating path and C is the composited colour at the pixel (an example is shown in Figure 1).

In our approach a pixel is extended to a net of $K_a{*}K_a$ cells as illustrated in Figure 2. (K_a can be any integer greater than unity. Generally the rendered image is finer if K_a is larger.) Then, for the colour accumulation process as illustrated in Figure 1, a transformation is made by distributing colours along the viewing direction over the $K_a{*}K_a$ cells. During the distribution process, each cell can be assigned a colour once. A brief description of the process is given by the following steps.

- Initially all the cells of the net are set to zero (vacant).
- First distribute the colour of the first sample (C_0, O_0) on

to the net. Let $stp_0 = 1/O_0$. Along the search path represented by SC (see Figure 2) a cell is given the colour value C_0 for every stp_0 vacant cells traversed. As a result, altogether $K_a{*}K_a{*}O_0$ cells are given the colour value C_0 with a uniform distribution on the net after all the cells have been traversed.

- Let $stp_1 = 1/O_1$ for the second sample (C_1, O_1). Traverse the remaining $K_a{*}K_a{*}(1 - O_0)$ vacant cells from the start along the same search path SC. A new vacant cell is given the colour value C_1 for every stp_1 vacant cells traversed. Consequently, altogether $K_a{*}K_a{*}(1 - O_0){*}O_1$ cells are given the colour value C_1 after all the vacant cells have been traversed, and these newly assigned cells are also distributed evenly on the net.
- The process is repeated until all n samples have been processed or all $K_a{*}K_a$ cells have been filled.

It is easy to prove that the above method for colour composition is valid. Without loss of generality, suppose a sample, say the $(j+1)$th one, has been processed. If we divide the number of vacant cells left at that moment, $K_a{*}K_a{*}\Pi_{i=0}^{j}(1 - O_i)$, by the total number of cells, $K_a{*}K_a$, the result $\Pi_{i=0}^{j}(1 - O_i)$ can be derived. We notice that this is the same as the value of the translucency after $j+1$ samples have been processed along the front-to-back path using equations (2) and (3). Finally, after the last sample (C_n, O_n) has been processed, $K_a{*}K_a{*}O_n{*}\Pi_{i=0}^{n-1}(1 - O_i)$ vacant cells are newly assigned the colour value C_n in a uniform distribution.

Lastly, if we sum up the colour values of all cells on the net and divide the summation by $K_a{*}K_a$, the result is then derived as

$$C = \sum_{j=1}^{n} C_j O_j \prod_{i=0}^{j-1} (1 - O_i) \qquad (5)$$

Obviously, this is the same as equation (4).

The search path in the above procedure could be different from that shown in Figure 2, but it makes no difference as long as the net cells can be traversed uniformly. In our approach, extending each pixel to a net of $K_a{*}K_a$ cells is equivalent to extending the screen plane $K_a{*}K_a$ times. We call the extended screen plane a *virtual plane*. With the help of the virtual plane the colour contribution of each sample along the traversal path for each ray is then represented over a two-dimensional area.

In previous work[22] we also proposed a method for combining projection methods with this colour composition approach. Besides, we also proved that this combination could produce images of different sizes simply by using various sizes of compression frames without distributing colours over the virtual plane again, as long as the colour-filling process can be executed in an even fashion. Thus, by producing a series of images, the computation is much less than in the conventional technique using equations (2) and (3) directly.

Tools for Selective Rendering

From the discussion in the previous section we see that for a voxel with its colour represented on the virtual plane an operation may be directly made on it without disturbing the voxels in front of it. Therefore selective parts can be operated locally for rendering. Here three tools are introduced to perform refinement, intensification and reduction functions based on the virtual plane.

By 'refinement' we mean that some parts of the volume (or the full volume) are further refined to reflect more details of the data. 'Intensification' involves enhancing the display of some parts in the rendered image to show more clearly some particular feature or components. In contrast, 'reduction' allows one to weaken some effect or hide certain components on some parts of the image.

According to the *over* operation, the magnitude for a given component to show in a rendered image is determined mainly by its opacity. Thus the functions of refinement, intensification and reduction can be realized through adjusting the opacities of voxels within the data.

Refinement

Consider a given sample A on a traversing ray and suppose its colour and opacity are C_A and O_A respectively. Then, after the colour distribution of sample A terminates, $S{*}O_A$ cells on the virtual plane will be assigned the colour C_A if the number of vacant cells left is S at that moment.

Without loss of generality, suppose sample A is replaced during the refinement by a series of samples labelled from front to back as $B_1, B_2, B_3, \ldots, B_i$, with their colours and opacities represented by C_{B_1}, O_{B_1}, C_{B_2}, O_{B_2}, C_{B_3}, O_{B_3}, $\ldots$, C_{B_i}, O_{B_i} respectively. Obviously, the composition results of colour and opacity of the new series of samples should be roughly C_A and O_A respectively. In the following we discuss the algorithm to calculate the results of the refined samples, and the procedure for rendering the refined image.

While sample B_1 is processed, its step for colour distribution is $stp_{B_1} = 1/O_{B_1}$. Thus $S^*O_{B_1}$ cells will be set with the colour C_{B_1}, because S vacant cells were left before sample A was processed. However, sample B_1 can take effect only in S^*O_A cells during the refinement; therefore the step for B_1 should be modified. Let the new step for B_1 be $newstp_{B_1}$. We have

$$S^*O_{B_1} = S^*O_A/newstp_{B_1}$$

$$newstp_{B_1} = O_A/O_{B_1} = stp_{B_1}{}^*O_A$$

While sample B_2 is processed, we have

$$(S - S^*O_{B_1})^*O_{B_2} = (S^*O_A - S^*O_A/newstp_{B_1})/newstp_{B_2}$$
$$newstp_{B_2} = stp_{B_2}{}^*(O_A - O_{B_1})(1 - O_{B_1})$$

Similarly, we may derive $newstp_{B_3}$ for sample B_3 and, in general, $newstp_{B_i}$ for sample B_i as

$$newstp_{B_3} = stp_{B_3}{}^*\left(O_A - 1 + \prod_{j=1}^{2}(1 - O_{B_j})\right)\Big/\prod_{j=1}^{2}(1 - O_{B_j})$$

$$newstp_{B_i} = stp_{B_i}{}^*\left(O_A - 1 + \prod_{j=1}^{i-1}(1 - O_{B_j})\right)\Big/\prod_{j=1}^{i-1}(1 - O_{B_j})$$

The derivation is as follows.

Suppose $newstp_{B_{i-1}}$ for sample B_{i-1} is

$$newstp_{B_{i-1}} = stp_{B_{i-1}}{}^*\left(O_A - 1 + \prod_{j=1}^{i-2}(1 - O_{B_j})\right)\Big/\prod_{j=1}^{i-2}(1 - O_{B_j})$$

Then, when sample B_i is processed, we should have the equation

$$\left(S^*\prod_{j=1}^{i-2}(1 - O_{B_j}) - S^*\prod_{j=1}^{i-2}(1 - O_{B_j})^*O_{B_{i-1}}\right)O_{B_i} =$$
$$S^*O_A{}^*(1 - 1/newstp_{B_1})(1 - 1/newstp_{B_2})$$
$$\ldots(1 - 1/newstp_{B_{i-1}})/newstp_{B_i} \qquad (6)$$

As it can be derived that

$$1 - 1/newstp_{B_{i-1}} =$$
$$\left(O_A - 1 + \prod_{j=1}^{i-1}(1 - O_{B_j})\right)\Big/\left(O_A - 1 + \prod_{j=1}^{i-2}(1 - O_{B_j})\right)$$

equation (6) can be written as

$$O_{B_i}\prod_{j=1}^{i-1}(1 - O_{B_j}) = \left(O_A - 1 + \prod_{j=1}^{i-1}(1 - O_{B_j})\right)\Big/newstp_{B_i}$$

Therefore

$$newstp_{B_i} = stp_{B_i}{}^*\left(O_A - 1 + \prod_{j=1}^{i-1}(1 - O_{B_j})\right)\Big/\prod_{j=1}^{i-1}(1 - O_{B_j})$$

From the above discussion we see that it is easy to obtain the new steps, as the computation could be conducted recursively. As for the refinement procedure over the virtual plane, the work procedure could be designed. The cells influenced by the original sample are first set to vacant, then the new samples distribute their colours one by one among these cells with their steps newly modified. This operation is performed only on the portion of the image in which users are interested.

Intensification

Consider a given sample A on a traversing ray and suppose its colour and opacity are C_A and O_A respectively. Then, after the colour distribution of sample A terminates, S^*O_A cells on the virtual plane will be assigned the colour C_A if the number of vacant cells left is S at that moment.

To enhance the effectiveness of sample A, we change its opacity to O_A^{new}, a value greater than O_A. Because the number of vacant cells left is S while distributing C_A, $S^*O_A^{new}$ ($>S^*O_A$) cells should be filled with the colour C_A to show the intensification. Apparently, the intensified sample A has to occupy some extra cells that were formerly filled by its following samples on the traversal path. Suppose the following samples are B_1, B_2, . . ., B_i from front to back, with colours and opacities C_{B_1}, O_{B_1}, C_{B_2}, O_{B_2}, . . ., C_{B_i}, O_{B_i} respectively. For sample B_1, $S^*(1 - O_A)^*O_{B_1}$ cells are filled with the colour C_{B_1} before sample A is intensified, but $S^*(1 - O_A^{new})^*O_{B_1}$ cells should be filled after sample A is intensified. Thus sample A should occupy the cells formerly owned by the sample B_1 with a step

$$[S^*(1 - O_A)^*O_{B_1}]/[S^*(1 - O_A)^*O_{B_1} - S^*(1 - O_A^{new})^*O_{B_1}] = (1 - O_A)/(O_A^{new} - O_A)$$

Similarly, the same step $(1 - O_A)/(O_A^{new} - O_A)$, independent of B_j, is derived for sample A to occupy the cells formerly owned by other samples B_j ($j = 0, 1, \ldots, i$).

Therefore intensification of sample A means occupying some cells formerly possessed by its following samples in

addition to its own cells. In this way the effect of the intensified sample A reflects over the virtual plane.

Reduction

Consider a given sample A on a traversing ray and suppose its colour and opacity are C_A and O_A respectively. Then, after the colour distribution of sample A terminates, S^*O_A cells on the virtual plane will be assigned the colour C_A if the number of vacant cells left is S at that moment.

To reduce the effectiveness of sample A, we change its opacity to O_A^{new}, a value less than O_A. Because the number of vacant cells left is S while distributing C_A, $S^*O_A^{new}$ ($<S^*O_A$) cells will be assigned the colour C_A after the reduction. By a similar derivation as discussed for the other two tools, the process to reduce sample A is realized by distributing its colour C_A among the S^*O_A cells, which were originally filled with C_A, by the step O_A/O_A^{new}. Obviously, the cells left empty owing to the reduction of sample A will be filled by its following samples on the traversal path. It is easy to prove that the steps for the following samples to fill the emptied cells do not need to be modified.

Application of Selective Rendering Tools

Integration with Ray-casting Approach

For the front-to-back ray-casting approach the tools discussed in the previous section can be easily integrated, except that each cell should record the sample that fills it. Thus each ray can conveniently take finer samples to composite colours by the refinement tool. Similarly, through the intensification and reduction tools the effect of samples can be intensified or decreased in the casting rays.

However, in practice, intensification and reduction may occur during a refinement process. If the accumulated opacity of the refining samples decreases as compared with that of the original sample, reduction is required after the refinement. Otherwise, if the accumulated opacity of the refining samples increases, the intensification operation is necessary during the refinement.

Thus a parameter is required for recording the accumulated opacity of the refining samples. As soon as the parameter exceeds the opacity value of the original sample, intensification takes place before the following refinement operations. The process continues until all the refining samples are processed.

To save computation, the operation of opacity accumulation for all the refining samples may be made before the refinement works.

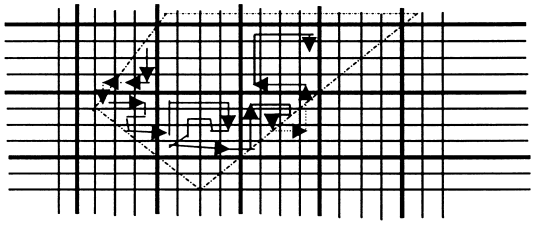


Figure 3.

Integration with Projection Approach

In the direct volume rendering by projection approach, such as perspective splatting and irregular volume rendering, the projection areas of samples on the screen plane may be of different sizes and overlap each other in a complicated fashion. This would surely bring a lot of difficulties in applying the tools. Fortunately, the problem can be easily resolved by adopting a strategy for designing the search path.

As we know, the virtual plane forms from the screen plane by extension. Each pixel corresponds to a net of $K_a^*K_a$ cells. Thus we may design the search path in such a way that, in the influence area of the sample, it goes through all cells corresponding to a certain pixel before it enters cells related to other pixels. Therefore the tools for selective rendering can be realized in a similar way to that introduced in the previous subsection. This is illustrated by an example in Figure 3. The grids formed by thin lines represent cells on the virtual plane, while the grids formed by thick lines express nets corresponding to pixels. The polygon formed by broken lines represents the influence area of a sample. The lines with arrows show the search path.

Suppose a virtual plane is extended from the screen plane with a pixel corresponding to a net of $K_a^*K_a$ cells. Then, when designing the search path, we make the assumption that a pixel corresponds to a net of $N_a^*N_a$ cells with $N_a<K_a$ so that a high-resolution image can be rendered without applying for extra storage. This is an exclusive advantage of the projection approach.

Implementation and Experimental Results

In the implementation of the proposed method a dynamic screen technique[27] is adopted to manage the virtual plane by storing linking information in cells. Therefore vacant cells can organize in links so that we can quickly know where to start filling up cells. The data structure for cells is

```
Struct { char red, green, blue;
         int object_num;
         char class_id;
         int nearx, neary;
         int hindx, hindy;
       } Virtual_cell;
```

In this data structure, *red*, *green* and *blue* refer to the colour filling in the cell; *object_num* is for recording which sample fills the cell; *class_id* expresses which classification the sample belongs to; *nearx* and *neary* are co-ordinates of the next cell that is also filled by the same sample along the search path; *hindx* and *hindy* are the co-ordinates of the first cell filled by the following sample of the current sample.

To test the efficiency of the new method, the sample buffer algorithm[16] is also implemented for comparison. This algorithm also provides the facility to visualize data selectively. It uses a list of sample buffers to store the intermediate results for each ray along its traversal path through volumes. Its advantage is to be able to save much time while producing a series of images in a refinement process. However, as the interesting parts in data are not definite, the *merging* operation of this algorithm is abandoned. The data structure for its sample buffers is

```
struct Samplenode { char red, green, blue;
                    float opacity;
                    int object_num;
                    struct Samplenode*sptr;
                  } Samplenode;
```

In this structure, *red*, *green* and *blue* refer to the colour of the sample occupying this node; *opacity* expresses the ratio for the sample to occlude rays; and *object_num* denotes the sample that occupies this node.

We tested a set of data of size 128*197*128 on a sloth body. To facilitate the implementation of the ray-casting approach, the data were compressed to the size 128*197*64 by averaging along a dimension prior to the test. In the test a coarse image is rendered in the compressed data to survey the data at first (step (a)). Then the reduction tool is used to remove all materials except the bone in the middle part of the body (step (b)). This is followed by an intensification step (step (c)) to show the detail of the heart. Finally (step (d)), the part in front of the heart is refined to show the ribs by using the original data. Figures 4(a)–4(d) were produced by the proposed method and Figures 5(a)–5(d) by the sample buffer method. All these images are rendered with shading in a parallel ray-casting approach. The enlargement ratio is 8 by 8 for setting up the virtual plane from the screen plane. The statistics of the time (in *seconds*) required for each step on an SGI Indigo[2] workstation are listed in Table 1.

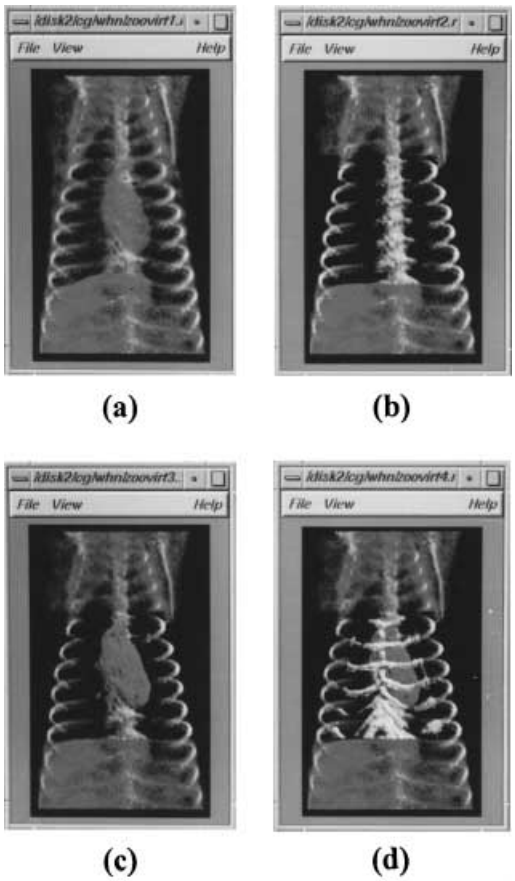


Figure 4.

The experimental results show that the proposed method is faster than the sample buffer method, especially for refinement. This is because the sample buffer method must conduct operations in 3D space though it saves the intermediate results while producing a series of images, and it also needs extra storage when new sample buffers

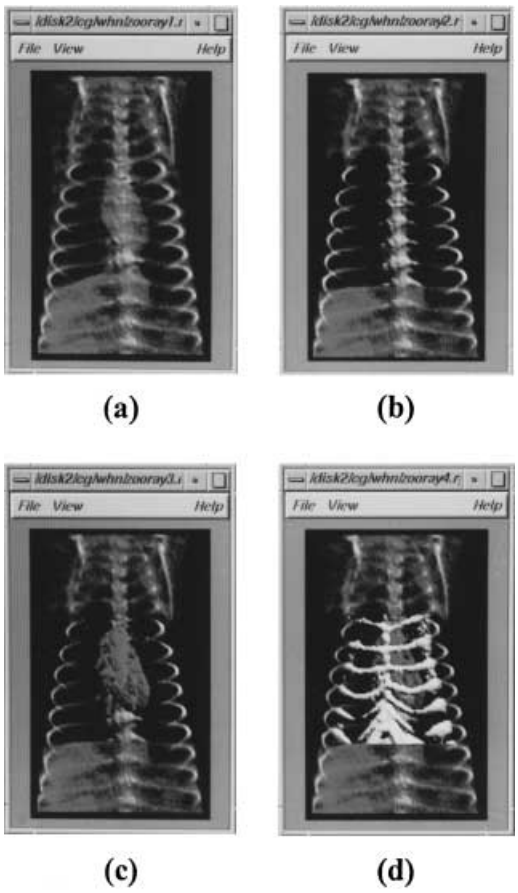


Figure 5.

are added during the refinement. As for the new method, it can operate directly on the interesting parts of the data without traversing those at their front. Furthermore, the iterative *over* operations along the accumulation path are reduced in the proposed method, by employing the compression operation on the virtual plane, and especially it has no requirement for new storage in the refinement process.

In this new method the virtual plane may demand much storage space. Fortunately, by investigating the experimental results, we know that, to produce images meeting normal requirements, it is unnecessary to set up a very large virtual plane. We can roughly compare these two methods implemented. Suppose the set of data is of size d*x**d*y**d*z* and each pixel casts a ray along the *dz*-direction in a parallel way. Then the new method needs d*x**d*y***rate***rate***sizeof(Virtual_cell)* bytes, where *rate* is the enlargement ratio for setting up the *virtual plane*. As for the sample buffer method, it needs d*x**d*y**d*z***sizeof(Samplenode)* bytes. The ratio between them is 1·5**rate***rate*/*dz*. In general, *dz* is not small and *rate* does not need to be very large. Thus the proposed method generally does not require more storage than the sample buffer method.

Conclusion

Selective visualization is useful in dealing with data of large size and dimensionality. A new method is proposed to accelerate the rendering to the interesting parts of data. By employing the virtual plane to store intermediate results and executing much work on the virtual plane instead of in 3D space, the new method can help scientists to quickly visualize whatever they are interested in.

The new method can integrate with all the current direct volume rendering techniques. In particular, when it combines with progressive refinement methods and selective methods, scientists can study data with high efficiency.

ACKNOWLEDGMENTS

This project is supported by the Natural Science Foundation of China.

References

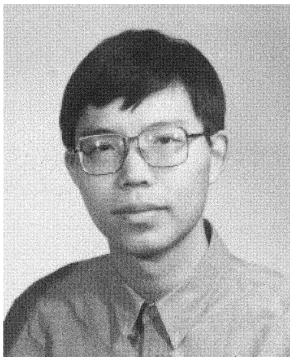
1. W. E. Lorensen and H. E. Cline, 'Marching cubes: a high resolution 3D surface construction algorithm', *Computer Graphics*, **21**(4), 163–169 (1987).

Method	Initial rendering	Reduction	Intensification	Refinement
Proposed method	319	70	8	153
Sample buffer method	340	104	18	388
Acceleration ratio	6·58%	48·57%	125%	153·59%

Table 1. Statistics of time (in seconds) for each step

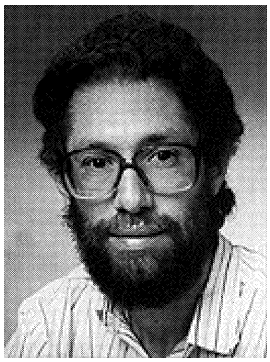
2. M. Fruhauf, 'Volume visualization on workstations: image quality and efficiency of different techniques', *Computers & Graphics*, **15**(1), 101–107 (1991).
3. J. Kajiya and V. H. Brain, 'Ray tracing volume densities', *Computer Graphics*, **18**(3), 165–174 (1984).
4. P. Sabella, 'A rendering algorithm for visualizing 3D scalar data', *Computer Graphics*, **22**(4), 51–58 (1988).
5. G. Sakas, 'Fast rendering of arbitrarily distributed volume densities' C. E. Vandoni and D. A. Duce (eds), *Proceedings of EUROGRAPHICS '90*, North-Holland, Amsterdam, 1990, pp. 519–530.
6. C. T. Silva and J. S. B. Mitchell, 'The lazy sweep ray casting algorithm for rendering irregular grids', *IEEE Transactions on Visualization and Computer Graphics*, **3**(2), 142–157 (1997).
7. W. Wang, D. Zhou and E. Wu, 'Accelerating techniques in volume rendering of irregular data', *Computers & Graphics*, **21**(3), 289–295 (1997).
8. R. Yagel and A. Kaufman, 'Template-based volume viewing' A. Kilgour and L. Kjelldahl (eds), *Proceedings of EUROGRAPHICS '92*, North-Holland, Amsterdam, 1992, pp. 153–167.
9. M. B. Haley and E. H. Blake, 'Incremental volume rendering using hierarchical compression', in *Proceedings of EUROGRAPHICS '96*, North-Holland, Amsterdam, 1996, pp. 45–55.
10. P. Lacroute and M. Levoy, 'Fast volume rendering using a shear–warp factorization of the viewing transformation', *Annual Conference of ACM SIGGRAPH '94*, 1994, pp. 451–457.
11. D. Laue and P. Hanrahan, 'Hierarchical splatting: a progressive refinement algorithm for volume rendering', *Computer Graphics*, **25**(4), 285–288 (1991).
12. N. Max, P. Hanrahan and R. Crawis, 'Area and volume coherence for efficient visualization of 3D scalar functions', *Computer Graphics*, **24**(5), 27–33 (1990).
13. P. Shirley and A. Tuchman, 'A polygon approximation to direct scalar volume rendering', *Computer Graphics*, **24**(5), 63–69 (1990).
14. L. Westover, 'Footprint evaluation for volume rendering', *Computer Graphics*, **24**(4), 367–376 (1990).
15. J. Wilhelms and A. V. Gelder, 'A coherent projection approach for direct volume rendering', *Computer Graphics*, **25**(4), 275–284 (1991).
16. H. R. Ke and R. C. Chang, 'Sample buffer: a progressive refinement ray-casting algorithm for volume rendering', *Computers & Graphics*, **17**(3), 277–283 (1993).
17. J. Wilhelms and A. V. Gelder, 'Octrees for faster isosurface generation extended abstract', *Computer Graphics*, **24**(5), 57–62 (1990).
18. J. Wilhelms and A. V. Gelder, 'Octrees for faster isosurface generation', *ACM Transactions on Graphics*, **11**(3), 201–227 (1992).
19. I. Ihm and R. K. Lee, 'On enhancing the speed of splatting with indexing' G. M. Nielson and D. Silver (eds), *Proceedings of IEEE Visualization '95*, IEEE Press, New York, 1995, pp. 69–75.
20. D. Silver, 'Object-oriented visualization', *IEEE Computer Graphics and Applications*, **15**(3), 54–62 (1995).
21. T. V. Walsum and F. H. Post, 'Selective visualization of vector fields', *Computer Graphics Forum*, **13**(3), 339–348 (1994).
22. W. Wang and E. Wu, 'A new method of compositing colors in volume rendering', presented at *Fifth Eurographics Workshop on ViSC*, Rostock, May–June 1994.
23. L. Sobierajski, D. Cohen, A. Kaufman, R. Yagel and D. E. Acker, 'A fast display method for volumetric data', *The Visual Computer*, **10**, 116–124 (1993).
24. J. K. Udupa and D. Odhner, 'Fast visualization, manipulation, and analysis of binary volumetric objects', *IEEE Computer Graphics and Applications*, **11**(6), 53–62 (1991).
25. J. K. Udupa and D. Odhner, 'Shell rendering', *IEEE Computer Graphics and Applications*, **13**(6), 58–67 (1993).
26. T. Porter and T. Duff, 'Compositing digital images', *Computer Graphics*, **18**(3), 253–259 (1984).
27. R. A. Reynods, D. Gordon and L. S. Chen, 'A dynamic screen technique for shaded graphics display of slice-represented objects', *Computer Vision, Graphics and Image Processing*, **38**, 275–298 (1987).

Authors' biographies



Wencheng Wang is an associate professor in the Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. His current research interests are scientific visualization, realistic image synthesis and virtual reality. He received his BS in computer software in 1988 from Xiangtan University and his MS and PhD in computer graphics in 1993 and 1998 respectively from the Institute of Software, Chinese Academy of Sciences.

Enhua Wu graduated from Tsinghua University, Beijing in 1970 and lectured at the university from 1970 to 1980. He conducted PhD research from 1980 to 1984 in the Department of Computer Science, University of Manchester, U.K. and received his PhD in 1984. Since 1985 he has been working at the Institute of Software, Chinese Academy of Sciences, where he became a full professor in 1992. In recent years he has been invited for visiting research by Lawrence Livermore National Laboratory, Utah State University, the Chinese University of Hong Kong and the University of Hong Kong. He has also been a visiting professor since September 1997 in the Faculty of Science and Technology, University of Macao. His main interests are computer graphics, CAD and computer simulation, including scientific visualization, physically based animation and virtual reality.



Nelson Max is a professor at the University of California, Davis/Livermore and a computer scientist at Lawrence Livermore National Laboratory. His research interests are realism in nature images, molecular graphics, computer animation and 3D scientific visualization. He served as computer graphics director for the Fujitsu pavilions at Expo 85 and 90 in Japan. Professor Max received his PhD in mathematics from Harvard University in 1967 and is a member of ACM Siggraph.