Sillion, F., Puech, C. 1989 A General Two-Pass Method Integrating Specular and Diffuse Reflection. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3):335-344

Wallace, J., Cohen, M., Greenberg, D. 1987 A Two-Pass Solution to the Rendering Equation: A Synthesis of Ray tracing and Radiosity Methods. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):311-320

Fractal Volumes

C. Kose, C. P Willis and D. J. Paddon
Department of Computer Science,
University of Bristol,
Queens Building,
University Walk,
Bristol BS8 1TR, UK.
email: derek@compsci.bristol.ac.uk

Abstract

A Mandelbrot set, in quaternion representation, is used as a test data set to research the necessary characteristics of a system that visualises large and complex data sets. An image space software solution is developed that uses dynamic data partitioning across a distributed system of processors. A system configuration is used that minimised the network diameter in order to reduce the communications overhead.

1 Introduction

Most volume visualisation is generated from scientific data because of the need to analyse the nature of the data. Examples of this type of work comes easily to mind. For example, the two-dimensional slices of data that are obtained from CAT or MRI scanners are best viewed as a three-dimensional stack of slices. In this form of visualisation the original set of slices can be viewed from an arbitrary position showing any desired spatial relationship.

In computer graphics most rendering problems are interested in surface representation. Once occluded surfaces, reflections and refractions have been established few if any three-dimensional effects remain to be solved. However, if the computed image is represented as a volume visualisation we can include effects such as opacity, thus extending the complexity of the representation of the foremost surface and even some of the occluded surfaces. Also, if the viewer interacts with the image to produce partially-transparent objects, then the entire volume of image data may be required to contribute to the final image.

There are three motivations for this research. The first is to extend the range of effects that can be incorporated in a synthesised image. The second is to reduce the high cost of computing the images by finding efficient algorithms. The remaining motivation is to reduce the time required to produce the images to an

acceptable level by rendering the image on multiple instruction multiple data computers (MIMD).

2 Volume Rendering Models

The solutions proposed in this research are software based and, therefore, are suitable for general purpose MIMD computers or systems of workstations. However, it is hoped that the results also give insight into methods needed for designing and building special purpose architectures for volume rendering and image synthesis.

Volume rendering is the term used to describe the operation of converting volume data into a visual material that consists of regions or layers of variable transparency and colour. The advantage of this rendering procedure for images that exhibit degrees of transparency is that much or all of the volume of image data is available to the observer, irrespective of viewing direction, without recourse to using artificial geometry that does not exists in the image.

We use an algorithmic approach that renders an image of a volume from an arbitrary viewpoint. We sample data on a regular grid, called the object lattice. To view the data we embed an image lattice which is aligned with the viewing direction. Thus, the data field values are known on the object lattice points, and therefore, the first stage in rendering an image is to calculate the data field values on the image lattice. That is, reconstructing the continuous three-dimensional data from the object lattice and resampling this data at the image lattice nodes. The second stage in the rendering procedure is to classify and shade the sampled data values on the image lattice. The final stage of the rendering process is to composite the image lattice data to give a final colour at each pixel in the image plane.

Three approaches to performing the rendering stages have been widely used in practice: volume shearing; splatting; and ray casting. for uniprocessor and some parallel implementations it has been shown that the most efficient rendering method is splatting, particularly so as the image size increases relative to the data size (Neumann 1993). However, in this work a number of factors lead us to the conclusion that ray casting is the more appropriate method for implementing the visualisation of complex data sets on MIMD computers. In the case of splatting and volume shearing it is necessary to carry out two-dimensional computations on the object lattice data. These operations are well suited to parallel computers that use synchronous cycles over a plane of processors, such as a single instruction, multiple data computer (SIMD) , for example, the connection machine. The ray casting algorithm (Levoy 1988. Sabella 1988) allows

us to take advantage of the inherent independent parallelism that is present at every pixel in the object and image planes. This parallelism, is therefore, asynchronous and ideally suited to a MIMD mode of computation.

We are interested in volume rendering problems which exhibit varying degrees of complexity through the object data. Therefore, we can expect that an efficiently processed data set will require varying amounts of processing for a given unit of volume space. The implication of this situation is that a uniform division of data across a set of processors is unlikely to lead to an evenly balanced processor load but instead gives an inefficient system. Thus the splatting and shearing algorithms are less suited to handling varying data complexities than the ray casting algorithm.

2.1 Ray Casting

Ray casting is perhaps the most commonly used method in rendering volume data. The principle is similar to ray tracing, that is, a ray is projected from the view point through every pixel in the image plane. Along each ray path the object data is sampled at regular intervals. Trilinear interpolation is used to reconstruct the field data at the lattice nodes. Then, the computation proceeds ray by ray until all pixel values in the image plane have been processed. The new sample points in the object plane are shaded, classified, and composited. The composition can be carried out in two ways: from back to front, by accumulating colour along the ray upto the sample point, when the data has low opacity; or front to back when the data has high opacity. In the latter case, contributions to the final accumulated colour can be terminated when the opacity exceeds a threshold, as further contributions are occiuded by the opaque material that has been previously accumulated. This method can save significant work when rendering opaque data sets.

This ray casting implementation uses octree data structures to accelerate the intersection tests of the ray and the lattice nodes. Volume data is pre-processed so that the octree leaves contain a description of the lattice nodes. As the ray traverses space, empty sub volumes are easily identified by the octree data structure. At sample points within a sub volume the octree is traversed to access the field descriptors.

2.2 A Fractal Test Problem

We have chosen Mandelbrot fractal volumes as a test problem for a number of reasons. First, the data sets allow us to set up parallel systems within which the parallelism of each pixel is independent, that is, without any computational data

dependencies. Second, the nature of Mandelbrot fractals is such that we can expect local complexities of data to vary widely in a manner that cannot be predicted. Thus, the object lattice may need local refinement if an accurate solution is to be obtained, and as the fractal objects have the property of supporting infinite resolution we are able to generate test problems that satisfy the requirements of the research programme. The need to locally refine the solution is, in principle, not unlike the solution that is obtained with a progressive refinement approach (Bergman 1986). The characteristics exhibited by fractal sets are therefore ideal for testing the MIMD rendering system that we propose.

The structure of fractal objects is commonly investigated in the complex plane. It is a simple extension to represent these three dimensional functions as four-dimensional quaternions. The representation of Mandelbrot sets are conventionally generated by assigning a colour to the value of each Z_0 of the complex plane

$$Z_n \leftarrow Z_{n-1}^2 + Z_0$$

on the basis of the number of iterations of this function that are required for the absolute value of Z_n to exceed some preset limit.

An alternative representation is as a quaternion. Then

$$Z_k = f(Z_{k-1}); \quad \text{with } Z_k = Z_0$$

where k is known as the periodic basin of attraction and gives the colour of the pixel (Hart 1993). In four-dimensions any point in the Mandelbrot set is represented as

$Q = a + bi + cj + dk$, where $i^2 = j^2 = k^2 = -1$ are the three imaginary components whose axes are perpendicular to the real axis as well as to each other.

Using conventional generation techniques, the four-dimensional quaternion is easily produced. This four-dimensional space is visualised by taking its intersection with the three-dimensional subspace of quaternion values. the three dimensional object lattice is then produced by grouping together a set of two dimensional slices.

Any point sample of the fractal object can be characterised by a colour C(R,G,B) and an opacity α. Thus a voxel based representation is easily built that represents object lattice nodes in terms of colour and opacity.

We use a lighting model that combines a pseudo-colour mapping and a local reflection model. Then we will be able to view the structure of the quaternion as a volume within which objects are nested within one another. These objects are projected as colours, densities and opacities. In the first instance, each voxel is given a (R,G,B,α) value in the first instance according to a classification procedure. Then we modulate the colour with an intensity function by applying a local reflection model such as the Phong reflection model. Surface detection is obtained by evaluating surface normals.

3 Parallelisation of the volume visualisation

Two important issues are immediately evident when decisions are being made about the system design. The first issue is the way in which data is managed and the second is the method of system configuration. These issues can, of course, have direct bearing on each other. As we are using Inmos Transputers to build the system we can apply a high degree of flexibility to our configuration design, within the constraint of having four interprocessor links available on each processor. We will first address the issue of data management.

3.1 Data model

When a volume rendering problem is solved on a distributed memory computer, two basic methods are available. The object data can be replicated across the system so that every process has identical data stored in its memory. Then, no interprocessor data communication is needed, lending to optimum load balancing across the system. From a practical standpoint, this is the objective that should be aimed for. We note that the cost effectiveness of this model may be low. From the research point of view, there are no issues that merit our attention as the implementation of such a system is trivial. However, when the available memory is insufficient to replicate the data, and a partitioning of the data is required, then the system gives rise to many research issues. We will study the problems associated with a partitioned data set.

There is ample evidence in the parallel computer graphics literature, over the past five years, that object space models are ideally suited to problems that are undemanding in terms of complexity of image or when ample memory is available for the problem size. However, when the image has local complexity then the data needs redistributing among processors. The redistribution of data in this object model has been shown to be inefficient (Green 1991). The alternative approach, that is an image model, is flexible and leads to efficient local balancing across the processors. (Green 1989, Priol 1989).

For relatively simple visualisation problems, it would be possible to adopt an image space model that uses statically partitioned data, with partitioned edges being overlapped or accessed from the appropriate processor. Again, we may have some load imbalance as any *a priori* estimate of data distribution may be inaccurate for problems exhibiting local variations in computational complexity. A solution to a system design that allows problems to be solved that exhibit severe variations in local computational complexity must adopt a dynamic data management strategy. Then, data migrates on demand to processors that are under-utilised, thereby evening out the load balance of the system.

This current approach to a visualisation system has the additional feature of allowing data sets to be handled that exceed the total distributed memory capacity. To date, the reported system designs solve problems strictly within the limits of the size of the distributed memory. Computers such as the Delta Touchstone are capable of solving problems with data sets in the low gigabyte range. However, problems exist that require terabytes of memory. It is unlikely that systems will achieve that size of memory for a considerable time. Therefore, the ability to dynamically distribute data will be essential if very large problems are to be solved.

By retrieving data from some central storage device, data can be distributed to processors by overwriting previously held data. Thus a dynamic image space data model is capable of being scaled to problems of arbitrary computational complexity and size of data sets.

3.2 Configuration model

The test problem we are using is one of a class of problems that enable data to be generated "on the fly" to enable a higher resolution image to be visualised. Similar computational effects are imposed on the system when coarse and fine visualizations are made from forward and reverse zoom actions on data sets. If we are not to impose artificial restrictions on our ability to resolve or coarsen visualisations, while maintaining efficient processor utilisation, then again, an image space solution with dynamic data management is required.

Most of the visualisation methods that have been reported in recent times have been artificially constrained to operated on a commercially provided system configurations. The majority of these configurations being grids of processors. The solutions chosen, such as static data distribution, or accessing near-neighbour data have usually been as a consequence of the inherent difficulty of using the two dimensional grid of processors in any other mode of operation. In this current research, we are not constrained by any system configurations and

are able to explore methods that help us to minimise communication overheads. Following work that has been ongoing since 1988 in parallel radiosity methods we have chosen to use a minimum path (AMP) configuration (Chalmers 1991, Paddon 1993).

For a given communications bandwidth, the total number of communication messages on a system and the density of these messages are a function of the network configuration. For problems with very large data sets, or those requiring large redistribution of data due to zoom effects, we can expect the communication characteristics of the system, and in turn the performance characteristics, to be directly related to the density of messages on the network. The density of communication messages is directly related to the diameter of the system. From the Table 1 (Chalmers 1991) we see that a hypercube-like configuration gives the smallest system diameter, whilst at the other extreme, we get the largest system diameter from linear or ring arrangements of processors.

	Processing Elements							
	8	13	16	23	32	42	53	63
AMP	2	2	3	3	3	4	4	4
Hypercube	3	-	4	-	5	-	-	6†
Torus	3	-	4	-	6	6	-	7
Ternary Tree	4	4	5	6	6	7	7	7
C. C. Cycle	4	-	6	-	7	-	-	-
Mesh	4	-	6	-	10	11	-	14
Binary Tree	5	6	7	7	9	9	10	10
Ring	4	6	8	11	16	21	26	31
Chain	7	12	15	22	31	41	52	62

†64-processing element hypercube

Table 1: Comparison of configuration diameters

Table 2 (Chalmers 1991) shows that the average interprocessor distance is clearly lowest when a system is configured as an AMP. The AMP configuration has a distinct advantage on interprocess average distance when compared to a mesh connected computer, at a ratio of nearly 1:2. This results in a lower communication message density with a corresponding increase in system performance.

	Processing Elements							
	8	13	16	23	32	42	53	63
AMP	1.28	1.55	1.74	2.06	2.31	2.55	2.77	2.92
Hypercube	1.50	-	2.00	-	2.50	-	-	3.00†
Torus	1.50	-	2.00	-	3.00	3.21	-	3.94
Ternary Tree	1.97	2.56	2.91	3.39	3.93	4.37	4.77	4.99
C. C. Cycle	2.00	-	3.06	-	3.84	-	-	-
Mesh	1.75	-	2.50	-	3.88	4.23	-	4.91
Binary Tree	2.20	3.03	3.44	4.08	4.89	5.50	6.08	6.48
Ring	2.00	3.23	4.00	5.74	8.00	10.50	13.25	15.75
Chain	2.63	4.31	5.31	7.65	10.67	13.99	17.66	20.99

†64-processing element hypercube

Table .2: Comparison of average interprocessor distances

A Minimum Path (AMP) system consists of two parts:

1. a number of processors arranged in a minimum path configuration; and,
2. system software for controlling input/output, communication, task allocation and data management.

The philosophy underlying the construction of a minimum path configuration is to minimise the diameter of the interconnection network; that is, to minimise the number of links a message has to travel between any source processor and any destination processor within the configuration. This principle is maintained even at the expense of the loss of regularity in a system. Another feature of the AMP configuration is that every link of a processor within the configuration is connected to a different processor; that is, no two links of any processor are connected to the same processor. The 32-processing element AMP, which has a diameter of 3, is shown in the figure.

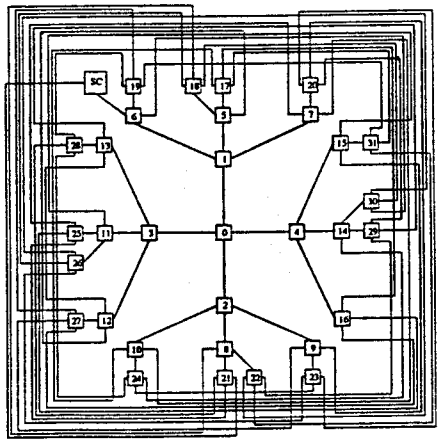

Figure The 32-processing element AMP configuration

3.3 Virtual shared memory

The distribution of data, or redistribution of data because of the needs of data coherence, requires that we impose a virtual memory management strategy on top of a hardware system that only allows local addressing of memory. The approach we take is to give data items a unique identity which then allows any processor to access that data via its identity (Green 1991). To enable each processor to directly address a data item, it is necessary for each processor to carry a table of identity locations. This is not feasible for large data sets, therefore, in general, the minimum information on the location of data is retained on each processor. The initial distribution of data is done as blocks of nodes, therefore, it is possible to maintain a table of the block distributions, and later subdivisions of the blocks. However, by ensuring that the boundary of data blocks are overlapped, it's unnecessary to maintain any information on the location of data for problems that are restricted to fixed or local refinement of the solution.

4 Conclusion

The solution of large data sets with dynamic local refinement are being researched on a system of processors that use an image space model with dynamic data management. The system configuration is arranged as a minimum path network of processors on which data is represented as a virtual shared memory model.

The initial work has enabled us to predict that the system is scalable to arbitrary size and complexity of problem. However, the efficiency of the inter-processor communications are critical to the effective scaling of the network of processors for a fixed size of data nodes. The organisation of data and its effective redistribution needs to be examined in detail before a full understanding of the system is achieved.

Bibliography

Bergman, L., Fuchs, H., Grant, E. and Spach,S. Image Rendering by Adaptive Refinement. Computer Graphics, 20(4), 1986, pp 29-37.

Chalmers, A. G. A Minimum Path System for Parallel Processing. PhD Thesis, University of Bristol, 1991.

Green, S. Parallel Processing for Computer Graphics. Pitman 1991.

Green, S. and Paddon, D. J. Exploiting Coherence for Multiprocessor Ray Tracing. IEEE Computer Graphics and Applications, 9(6), November 1989, pp 12-26.

Hart, J. C., Lescinsky, G. W., DeFanti, T. A. and Kauffman, L. H. Scientific and Artistic investigation of multi-dimensional fractals on the AT & T Pixel Machine. The Visual Computer 9, 1993, pp 346-355.

Levoy, M. Display of Surface from Volume Data. IEEE Computer Graphics and Applications, 8(3), 1988, pp 29-37.

Neumenn, U. Parallel Volume Rendering Algorithms Performance on Mesh-Connected Multiprocessors. Proceedings of Siggraph 1993 Parallel Rendering Symposium, San Jose, 1993. pp 97-104.

Paddon, D. J. and Chalmers, A. G. The Effect of Configurations and Algorithms on Performance. In Parallel Computing on Distributed Memory Multiprocessors, NATO ASI Series, Springer-Verlag, 1993, pp 77-97.

Priol, T. and Bouatouch, K. Static Load Balancing for a Parallel Ray Tracing on a MIMD Hypercube. The Visual Computer, 5(1/2), 1989, pp 109-119.

Sabella, P. A Rendering Algorithm for Visualizing 3D Scalar Fields. Computer Graphics 22(4), 1988, pp 51-88.

VISUALIZATION OF THE VOLUME DATASETS IN SCIENCE

Juraj Jankovič
Faculty of Mathematics and Physics
Comenius University
842 15, Bratislava, Slovakia

Volume visualization is one of the fast-growing areas in scientific visualization. It helps to look at scalar or vector datasets and understand them more easily. An overview of basic algorithms used for this purpose, some of enhancements and optimizations are described here.

Introduction

This paper contains a summary of the most important contributions to visualization of volume data. The earliest techniques are mentioned together with more detailed descriptions of the well-known algorithms, which are already used in science, and the newest enhancements.

The first part explains the terms, procedures and heuristics used in the field. This is important, because the standard language is still not established for this field. The main part covers volume rendering techniques and algorithms, their advantages and disadvantages. More space is devoted to the most important methods, but it is impossible to cover everything in one paper. So it is necessary to consult the papers listed in bibliography for detailed information.

Data

Because the large number of fields of science, which produce the volume datasets, is increasing all the time, together with the new applications and techniques, the volume data can be obtained from different sources. The basic categories of these sources were identified in [Watt92] :

1. Empirical data sets constructed from a mathematical model such as Computational Fluid Dynamics (CFD).

2. Data sets derived from an object, such as by tomographic scanning in the medical area.

3. Data set is some kind of computer graphics model.

There are many techniques available today. For the second category for example - Computed Tomography (CT), Magnetic Resonance Imaging (MRI), Positron Emission Tomography (PET), Single Photon Emission Computed Tomography (SPECT) - are all used in medicine. These are the methods used mostly in science. However, there exist some another ways of acquiring volume data [Elvi92] : voxelizing geometric description of objects, sculpting digital blocks of marble, hand-painting in three-dimensions with a wand, or writing programs that generate interesting volumes using stochastic methods.

The algorithms described here deal with visualizing single scalar data volumes. But there are vector, tensor, bi-modal, and higher-dimensional data as alternatives to scalar data, which are used in different fields of science. Some of the techniques can be extrapolated to visualize these types of data, although visualization of most of them is still an active area of research.