A Parallel Radiosity System for Large Data Sets

David Stuttard, Adam Worrall, Derek Paddon and Claire Willis

Department of Computer Science
University of Bristol
Bristol, BS8 1TR, UK
derek@compsci.bristol.ac.uk

The radiosity method gives realistic results for synthesising images based on diffuse light interaction; however, the method is also very computationally expensive. Here we present a system for parallelising a progressive refinement radiosity method using a ray casting approach. The system decomposes the environment into two distinct entities, the model geometry and the radiosity sample points. These are then distributed amongst the processors and message passing communication protocols and caching methods are introduced to enable the system to cope with very large models while maintaining a high processing rate.

1 Introduction

One of the most important methods in realistic image synthesis at the moment is radiosity. However, even with the recent advances made in radiosity algorithms [10], producing a radiosity solution is still extremely computationally expensive. For many applications, such as the use in the prediction and evaluation of lighting environments, the need for quick feedback is essential. Using radiosity for this type of problem is difficult, even on state of the art hardware, since practical data sets are of great complexity. Although methods have been developed to deal with very large data sets [13], the time taken to generate a solution is often too great to allow such solutions to be used as part of the design process.

We use distributed memory parallel processing to deal with both of these issues; this will allow very large data sets to be processed, and also to be processed in accelerated time. [1]

This system was based on CARM, a radiosity system that has been developed at Bristol. CARM is based on progressive refinement, using ray casting [14] to calculate form factors. It also provided initial mesh processing, and adaptive subdivision of mesh elements. An octree data structure is used to accelerate the ray casting.

1.1 Overview of CARM and Previous Work

One definition of radiosity is as an attempt to generate sufficiently accurate (both numerically and visually) representations of the radiance functions that describe the surfaces of an environment. In this paper we will explore this definition and define a practical working system.

Most radiosity methods use a discretisation approach, in which the environment is divided into a finite set of patches. This approach was initially used in thermal engineering and applied to diffuse reflection by Goral et al [6]. The radiance function of each surface is approximated by the set of patches attributed to that surface. More recent methods have used polynomials [15] and wavelet models [7] to represent the radiance functions over surfaces.

The CARM system is patch-based, with each patch holding a mesh of sample points. Although discretised systems have a number of shortcomings (such as aliasing) they can accurately represent very abrupt changes in the radiance function. The input geometry to the system is triangulated; this allows all object primitives and also all complex objects to be represented by meshes. The input triangles are then automatically tested for D0 discontinuities (i.e. tests for intersecting or abutting triangles) and also for excessive size, and automatically subdivided if required. The triangles left after such subdivision correspond to a traditional hand-crafted radiosity patches.

Radiance values (or flux values) are stored at sample points, which are located at patch vertices, and also within the patch. These sample points are stored in a mesh, which can be triangulated and then interpolated to reconstruct the radiance function. By adaptive subdivision of the mesh the system allows sample points to be added as the solution progresses, When new sample points are added the mesh is re-triangulated; these triangles are analogous to traditional elements. Subdivision is controlled by examining the gradient and difference between neighbouring sample points; if either of these values exceeds a threshold, then a new sample point is introduced. A final check is made for the linearity of the radiance function between the neighbouring sample points; if the gradient is linear, no new sample points are created, since interpolation between the existing sample points is sufficient to model the function. Note that the new sample points are similarly checked.

Progressive refinement [2] can be seen as each patch shooting its energy out to the rest of the environment. This has the advantage that useful results can be generated quite early on in the solution, with most of the important energy transfers having been performed within the first few iterations. Progressive refinement has been shown to be equivalent to Southwell Relaxation, which, unfortunately, has a slow convergence rate, therefore, if the solution of the progressive refinement method is taken to full convergence then costly computation needs to be carried out.

One of the most recent algorithm to be applied to radiosity is hierarchical radiosity [10]. This method uses a rigorous error analysis to approximate form factors where such approximations are of the same order as other approximations made in the solution. The effect of this is to approximate whole blocks of the form factor matrix to just one form factor, resulting in fewer form factors having to be computed. In fact, the algorithm runs in O(n) time (with n being the number of elements), as opposed to O(mn) (with m being the number of patches), as in other solutions. However, hierarchical radiosity requires an initial linking stage to determine which blocks of form factors can be approximated; this involves visibility calculations, and refinement of the patches into elements where necessary. This linking information also has to be stored. Hierarchical radiosity works most efficiently when few initial patches are refined into a large number of elements; the technique is less effective for environments with complex initial geometry.

Although there are some penalties in using the progressive refinement method on straightforward scenes, the method has unique advantages over both standard radiosity and hierarchical radiosity when volatile scenes are being manipulated. Small changes and movement of objects in a scene can be dealt with on a local basis by the progressive refinement method while other methods need to maintain a global solution.

In CARM form factor calculation are based on a method given by Wallace et al [14]. Energy is shot from a source patch to receiving sample points, with the patch approximated to a disc, the vertex to a differential area, and the distance between the two assumed to be large compared to the size of the disc. This assumption is often broken, for example at corners, then analytical extensions are used.

We use ray casting to determine visibility, with an adaptive technique to allow a trade-off for time versus accuracy. Ray tracing has been accelerated by with an octree data structure, which uses the Samet algebraic method of traversal [12] as opposed to the standard approach of Glassner [5]. The Samet method climbs the octree as far as necessary and then descends a mirror route to reach the neighbour, whereas the Glassner method descends from the top of the octree, selecting the most appropriate child until a leaf is reached. The Glassner method is prone to floating point error, and also becomes inefficient as the tree increases in size, whereas the Samet method has an average case performance independent of the size of the octree.

Discontinuity meshing for first and second order discontinuities [11] is currently under development. In the region of discontinuities higher order technique generates radiance function meshes of greater accuracy than those produced by the adaptive subdivision strategies.

2 Parallel Model

The parallel model used in this system utilises a message passing architecture [4, 8, 9]. Each processor in the network has a small amount of memory and four serial links to other processors in the network. The processor (Inmos T800 Transputers) has hardware support for context switching allowing the use of several processes on one processor. This process model is usually known as "threading". The system uses threads to support several tasks on one processor. This leads to two considerations to be made about the parallel network; intra-worker communication and topology and inter-worker communication and topology.

2.1 Intra-worker communication and topology

Each processor in the network is of exactly the same form. Several processes are executed concurrently on each processor. There are four main parts. The Work Distribution Manager (WDM), the Database Manager (DBM), the Worker Threads (WT) and the Worker Processes (WP). There can be any number of worker threads up to a pre-defined maximum. As the processor has hardware support for time slicing, these threads are almost free in terms of extra processing time. The use of these threads means that several distinct tasks can be initiated at the same time, the processor then automatically switching to threads that are not waiting for data. This has the distinct advantage over explicit task control of very low overheads. However, allowing the processor exclusive control of task management by using its inherent hardware context switching has several drawbacks. The threads tend to synchronise into a "request data, do work" rhythm giving a strobing effect. In order to overcome this problem the system uses a combination of explicit task control and threads.

Tasks are initiated by the WDM along with a series of threads. The tasks are started are high priority and the threads at low priority. Only one task is running at any one time. When a task requests data which is not immediately available (i.e. the data is on another processor), it is de-scheduled. When no task is running, then the low priority threads are automatically time-sliced in to utilise any waiting time the processor may have. This combination of explicit task control and threads enables the highest utilisation of a processor.

The WDM has exclusive access to the rest of the network via the processor's external links. Any requests to the network for data must go via this process. This process is also responsible for the initiation of the Worker Threads and management of Work Processes. The DM controls all the data for the node.

2.2 Inter-Worker Communication and Topology

Inter-worker communication is a more difficult problem to solve. Bi-directional communication must be supported by the WDM of each processor. It is inevitable that at some point a processor will be sending data or task information to another processor which is at the same time sending a request. A traditional solution to this problem is to buffer the messages. This approach only succeeds in postponing the same problem until the buffers are full. We enforce a strict protocol so that the deadlock situation cannot arise. The topology used in the system is a tertiary tree.

The system can cope with very large data sets by virtue of an off-line backing store in the form of a hard disk on the root node of the tree.

3 Distribution of Geometry Data

The form factor calculations employ a ray-casting approach for visibility determination [14], therefore an octree spatial subdivision [5] system is used to accelerate the calculation. This means that there are two distinct structures that have to be maintained; the hierarchy of surfaces and patches, and the octree, which is essentially an index into all the sets of patches.

Radiosity models generally tend to have large data-sets. The reason for this is that realistic lighting simulations are usually performed on real-world type environments. An average size for a model to undergo radiosity treatment will have several thousand polygons. Most networks of parallel processors have nodes with a few megabytes of memory and consequently it is impossible to contain the entire geometry of the model on a single node especially if one has to retain sample points. It is primarily for this reason that the geometry data in this system is distributed through all the processors.

The geometry data is not modified by the radiosity process; the volatile data in the system is the radiance function approximations, which appear as meshes of sample points which fit over the corresponding patches. Since the geometry data is not updated, there is no requirement to ensure data consistency via uniqueness, as is required by sample points. This allows the geometry data to be duplicated over a number of processors.

Each processor will contain a cache of geometry data, which will be used to feed the form-factor calculations. If the data is not present in the cache, then it will have to be fetched from another processor. In the case of very large model databases this data may not reside on any processor and is fetched from disk by a central controller.

The other important issue of the storage of geometry data is that of how to implement the octree. In traditional uniprocessor systems, a leaf node in the octree will contain a list of pointers to patches, as well as pointers to its parent and children. However, this use of pointers is not possible in a distributed memory machine, and so another approach has to be used.

The first option is to use a naming scheme, in which each patch (and also each voxel) is given a unique name. Then, a voxel can contain the names of the patches which are relevant, and the cache can be partitioned into two areas, one for patches and another for voxels. If the required patch/voxel is not in the cache, then a request is put out as before. The main drawback of this technique is in the increased amount of network traffic it will generate; if a voxel is fetched from another node, and contains several patches, these patches will also be required. For n patches, this will require a further n data requests.

The second approach is to embed the patch information in a voxel. This makes the voxel much larger (in terms of memory), but more importantly makes the size of a voxel dynamic; empty voxels will be far smaller than voxels which contain a number of patches. However, it has the advantage of reducing the number of data requests made, at the expense of making the message size larger.

4 Distribution of Sample-Point Data

The ray casting approach in CARM means that we are tracing rays from a source patch to a sample point in the geometry of the model. In the normal radiosity model, the sample point is part of the geometry of the model, usually a polygon of some sort. As we are dealing with points in this system, they are distinct from the geometry, and as such are distributed amongst the parallel processing elements as separate entities. As was mentioned earlier, the geometry is simply broken down into voxels and distributed anywhere in the network with duplication allowed if there is space. However, if this was done to the sample points in the system, then when the flux values at a sample point are updated, all similar sample points in the system must also be updated. This would mean broadcast updates of all sample points some of which may be being used for other source patch sample point shoots. This is difficult, very expensive to resolve and increases the communication overhead. Therefore, sample points are distributed through the system with no duplication allowed.

4.1 Sample point distribution.

In the initial version of the radiosity system, sample points were indexed using a hashing strategy based on the Cartesian world co-ordinate position and normal of the sample point. In the parallel model, we also index the sample points, but instead of giving a location in a hash table, we use this value to give us a processor number. This number is then used to send a message to interrogate the processor suspected of containing the desired sample point. If it does contain the requested data then the relevant information is passed back to the calling processor. In some cases, sample points will not be placed on their first choice of processor, for reasons of space or sometimes due to load balance issues. In this case, a re-hash of the sample point is done at the target processor, and another message sent to the second choice, and so on. A return message is then sent to the original processor, along with the eventual target address when the sample point is located. In this way, the sample points are distributed evenly throughout the network

4.2 Load balancing with sample points.

One requirement of a well engineered radiosity system is the need for element subdivision in some areas of the scene, such as around lights or near dark shadow areas. Large numbers of sample points may be created in small areas of the geometry. The hashing strategy should be able to hash these values to different processors, but if it cannot, and large numbers of sample points are clustered on one or a few processors, then load balancing must be achieved. This can be done very simply by forcing a re-hash on some of the sample points. In this way, even though there may be space for the sample point in question on its first choice of processor, it is moved onto the re-hash choice.

It is intuitively obvious that the effectiveness of the hashing function directly affects the number of re-hashes needed, and consequently the number of messages needed in order to hit the desired sample point. However, the subdivision of the model and creation of new sample points also affects this. In the worst case, we may need to search all nodes in the network for a particular sample point.

5 Shooting Patches.

Shooting a patch P is the process of distributing the energy "belonging" to P out to the rest of the environment. The algorithm presented in Wallace [14] is employed; for every sample point S in the environment, an approximation to the form factor is determined, and the energy is transferred. Finally, the 'unshot' energy total for P is set to zero.

The form factor is approximated by using the analytical result for a disc imposed on a differential area. To account for occlusion, a single ray can be cast from S to a point on P. To test for partial occlusion, a number of rays can be cast to various points on P, and an approximation made to the degree of occlusion.

5.1 Determining Visibility

Determining visibility introduces several problems for the parallel system. By far the largest amount of communication in the system is caused by the need to determining if a ray from a source patch to a sample point is occluded. This procedure reduces down to a simple ray traversal of the octree. The octree is distributed across the network with duplication allowed. When a ray is spawned from a shooting patch, it is first traced into an initial voxel. A request for this voxel is initiated by the data manager on the node. If the information is present on that processor, then the data is returned to the task in order to determine visibility. If the data is not on the processor, then a request is sent to the network.

5.2 Use of caching in octree data

The system utilises a caching strategy for the octree geometry data. This means that data from other processors in the network can also be held in a dynamic cache on the processor requesting the data. Thus, for a particular source patch shooting to a cluster of sample points, there will be a certain amount of coherence for the data required. Since the system also uses threads, the same data may well be requested at the same time thus also reducing the message density. The cache currently used is a simple fully associative cache with a temporal replacement scheme. The octree data structure and message packets in the system consist of the voxel itself and all geometry contained therein. Consequently, voxels are smaller than would normally be used in a standard ray tracing problem.

6 Dynamic Load Balancing With Distributed Control

There are three main problems for load balancing in this system, as described below:

6.1 Sample Point Distribution

Since sample point data is dynamic, the data must be balanced so that it is not all concentrated on one processor (which may happen if subdivision occurs in one very small area in the scene). The method used to initially distribute sample points and nominally create new ones is detailed in Section 4. Re-distribution of "hot spots" is easily achieved by re-hashing several of the sample points when a certain tolerance level is reached. As the memory becomes more saturated then this tolerance for re-hash redistribution necessarily has to be increased.

6.2 Geometry Data Distribution

There is little load balancing needed for this data structure. The data is initially distributed through the system and onto disc backing store if needed. Dynamic caches store relevant data for a particular node. Since these caches are dynamic and data in them can be destroyed then the issue of load balancing need not be considered.

6.3 Task Balancing

The distribution of tasks through this system is initially fairly simple and is done on a random basis. As subdivision occurs, task "hot-spots" may appear due to higher concentrations of sample points being formed. Radiosity is unpredictable in that a-priori, one does not know where subdivision will occur due to shadows and other lighting artefacts.

Due to the unpredictable nature of the system, tasks can potentially cluster on specific processors. These tasks are redistributed by using a simple poaching strategy in order that processors are used to their full capacity. When a processor runs low on tasks to perform, it informs its neighbours. These then pass extra tasks back to the calling processor. In this way, tasks are smoothed over the network. As iterations in the progressive refinement process are performed, task creation becomes less of a problem as the solution stabilises and tasks are distributed evenly across the network.

7 Patch Subdivision

Without pre-processing, the initial distribution of sample points throughout the environment will not be sufficient to accurately model the radiance functions of the surfaces in the environment. As the solution progresses, new sample points have to be created, and the meshes updated to include them. Discontinuity meshing [11] is currently the best method to determine where to place new sample points; however, it is computationally expensive.

Another method for increasing the accuracy of the sample point mesh is adaptive subdivision [3]. After a patch has been shot, it examines all patches in the scene; if any fail a certain criterion (usually their radiosity gradient being too high) then more sample points are added. Although this method does not model the radiance functions as well as discontinuity meshing, and also creates more sample points, it operates much faster. Care has to be taken when selecting the values of the tolerances for adaptive subdivision; inappropriate values can cause too little or too much subdivision.

This automatic subdivision has an impact on the parallel solution to radiosity. Neighbouring sample points may be able to benefit from image coherence, that is they will probably require the same data for object intersection tests. Since sample points on the same patch are likely to be close, it is desirable to keep such sets of sample points together on the same processor, so that they can all use the data that has to be fetched. However, this can adversely affect the load balancing of the system; several initially sparse patches could quite quickly contain the majority of the sample points in the system, which would leave other processors idle. One solution to this problem is to distribute the sample points randomly throughout the system, but this would lose the benefits of coherence, and therefore vastly increase the number of off-processor data-fetches. Another solution is to divide patches when they contain too many sample points; then one of the new patches can be passed to another processor for load balancing. This preserves the coherency, and allows dynamic load balancing, but at the cost of making the geometry data volatile. Geometry data is only used for intersection tests, it is not necessary to broadcast updates to ensure that all duplicates remain consistent. This is because the area of the original patch is all that is required for intersection tests.

8 Rendering of Scenes

Currently, all rendering is done using a SGI Iris Indigo Elan. The object database is updated from the parallel network via the ethernet. This system is slow but produces reasonable results. It is anticipated that very large models could be visualised using this system, the database on the rendering engine being supplied from a super-set of data on the parallel network. In this way, super-large walk-through type models can be visualised in reasonable time.

Many applications at this time for radiosity and real time systems are of the architectural walk-through variety. For very large models the environment is usually divided up into rooms and corridors. Visibility determination from rooms can cut down the amount of the model that is needed for real-time rendering [13]. As the parallel system calculates an iteration of the radiosity solution, it also determines which part of the model needs to be rendered and provides the current data-set to be rendered to the rendering engine. In this way, the renderer only processes a portion of the entire model and therefore can provide much more interactive rates.

The efficiency of the system is being evaluated and improved by establishing the parameters needed for judging the ratio of tasks to threads that are required to reduce processor wait time to zero. The optimum size of a system in terms of processing elements and system configuration is known to be dependant on the scene complexity and also on the size of the data set. Parameterising this aspect of efficiency is a difficult problem which requires the evaluation of cache coherence strategies, message passing protocols, the original decomposition of the problem and its representation in an octree data structure. Active research is currently being undertaken to find solutions to this problem.

9 Conclusion

A basic radiosity system has been developed that produces high quality images from scene data of varying complexity. Through the combination of discontinuity meshing, adaptive subdivision of meshes and ray casting for calculating the form factors, the system is able to correctly render shadow areas, highlight regions and geometrically complex surfaces.

To accelerate the system to cope with real time changes to a radiosity scene the system has been parallelised and implemented on transputer processing elements. This has enabled the data sets to be expanded in size and complexity and also to be able to return solutions in a shorter period of time. The development of the system is continuing with the aim of determining the parameters which optimise individual processor element performance and also the overall system performance.

References

1 Chalmers A. G., Paddon D. P. (1989) *Parallel Radiosity Methods.* Proceedings of the First International Conference on the Application of Transputers. Liverpool.

2 Cohen M. F., Chen S. E., Wallace J. R., Greenberg D. P. (1988) *A Progressive Refinement Approach to Fast Radiosity Image Generation* Computer Graphics 22 (3), 75-84.

3 Cohen M. F., Greenberg D. P., Immel D. S., Brock P. J. (1986) *An Efficient Radiosity Approach for Realistic Image Synthesis* IEEE Computer Graphics and Applications 6 (2), 26-35.

4 Feda M., Purgathofer W. (1991) *Progressive Refinement on a Transputer Network.* Proceedings of the Eurographics Workshop on Rendering, Barcelona.

5 Glassner A. S. *Space Subdivision for Fast Ray Tracing* (1984) IEEE Computer Graphics and Applications 4 (10), 15-22.

6 Goral C. M., Torrance K. E., Greenberg D. P., Battaile B. (1984) *Modelling the Interaction of Light Between Diffuse Surfaces* Computer Graphics 18 (3), 213-222.

7 Gortler S. J., Schröder P., Cohen M. F., Hanrahan P. (1993) *Wavelet Radiosity* Computer Graphics 27 (4), 221-230.

8 Green S., (1991) *Parallel Processing for Computer Graphics.* Research Monograph in Parallel and Distributed Computing. Pitman.

9 Green S., Paddon D. J., Lewis E. (1988) *A Parallel Algorithm and Tree Based Computer Architecture for Ray Traced Computer Graphics.* Proceedings of Parallel Processing for Computer Vision and Display, Leeds.

10 Hanrahan P., Salzman D., Aupperle L. (1991) *A Rapid Hierarchical Radiosity Algorithm* Computer Graphics 25 (4), 197-206.

11 Lischinski D., Tampieri F., Greenberg D. P. (1992) *Discontinuity Meshing or Accurate Radiosity* IEEE Computer Graphics and Applications 12 (6), 25-39.

12 Samet H. (1989) *Neighbour Finding in Images Represented by Octrees* Computer Vision, Graphics and Image Processing (46) 367-386.

13 Teller S., Fowler C., FunkhouserT., Hanrahan P. (1994) *Partitioning and Ordering Large Radiosity Computations* Computer Graphics 28 (4), 443-450.

14 Wallace J. R., Elmquist K. A., Haines E. A. (1989) *A Ray Tracing Algorithm for Progressive Radiosity* Computer Graphics 23 (3), 315-324.

15 Zatz H. (1993) *Galerkin Radiosity : A Higher Order Solution Method for Global Illumination* Computer Graphics 27 (4), 213-220.