PARALLEL IMPLEMENTATION OF A GLOBAL LINE MONTE CARLO RADIOSITY

Roel Martínez, Adrià Forés and Ignacio Martín
Institut d'Informàtica i Aplicacions, Universitat de Girona, Campus de Montilivi, E-17071 Girona, Spain

Keywords: Global illumination, Radiosity, Monte Carlo.

Abstract: Radiosity methods are known by their expensive computational cost. To compute high quality images with a lot of polygons or patches may take hours. For this reason parallel processing will be a good option in order to decrease the computational cost. On the other hand, Monte Carlo methods offer good alternatives for parallelization, given their intrinsic decomposition properties in independent subtasks. We have implemented our multipath method for radiosity using a cluster of PCs. Results are presented for 1 to 8 processors, exhibiting a good efficiency and scalability.

1 INTRODUCTION

Global illumination algorithms compute the single reflection of the light many times to simulate the multiple reflections. To obtain a single reflection at a point the estimate of the incoming radiance from a direction should be weighted by the probability of the reflection to a given direction and integrated taking into account all the possible incoming directions. Consequently, global illumination is basically a numerical integration problem. Thus global illumination is computationally very expensive, which means that in order to obtain a high quality image, with a lot of polygons or patches, may take hours. This high cost makes researchers in the area to look into parallelization alternatives to reduce it (Reinhard et al., 1998).

Local Monte-Carlo approaches sample the domain of the integration randomly using a probability density p, evaluate the integrand $f(\vec{x})$ here, and provide the f/p ratio as the primary estimate of the integral. This estimate is accurate if we can find p to mimic f precisely, i.e. to make f/p as flat as possible. This strategy, which is commonly referred to as importance sampling, places more samples where the integrand is high. Since in practice p can be very far from the integrand, the estimator may have high variance. Thus to get an accurate result many independent primary estimators should be used and compute the secondary estimator as their average.

Global Monte-Carlo methods do not rely on finding good sampling density. Instead, they take advantage of the fact that it is usually easy to evaluate the f at a well structured set of sample points $x_1, \ldots, x_n$. The emphasis is on that the simultaneous computation of $f(x_1), \ldots, f(x_n)$ is much cheaper that the individual computation of $f(x_1)$,..., and $f(x_n)$ by a local method, thus in this way we can have many more samples for the same computational effort. The bad side of this technique is that, finding a probability density that simultaneously mimics the integrand at many points is very difficult, thus practical methods usually use uniform sampling probability. In this way, global methods are implemented using global uniformly distributed lines, first used in (Buckalew and Fussell, 1989), in contrast with "local" lines, generated from sampled points in the scene.

The multipath algorithm for radiosity (Sbert et al., 1996; Sbert, 1997) is part of this family of global lines algorithms which has seen a further development in (Besuievsky and Pueyo, 1997; Szirmay-Kalos and Purgathofer, 1998; Szirmay-Kalos, 1999; Bekaert, 1999; Martínez, 2004).

On the other hand, task farm (or naive) Monte Carlo parallelization is based on the fact that we can decompose a Monte Carlo computation with n lines or rays into m independent smaller ones with n/m rays each with no loss in precision (Zareski et al., 1995; Sbert et al., 1995; Alme et al., 1998).

We have implemented our parallel solution on a cluster of PCs. We choose a high performance cluster for decreasing the execution time of the radiosity algorithm.

MartÃnez R., ForÃ©s A. and MartÃn I.
PARALLEL IMPLEMENTATION OF A GLOBAL LINE MONTE CARLO RADIOSITY.
DOI: 10.5220/0001801501640169
In *Proceedings of the Fourth International Conference on Computer Graphics Theory and Applications (VISIGRAPP 2009)*, page
ISBN: 978-989-8111-67-8

The rest of the paper is organized as follows. In section 2 we present the radiosity multipath method. In sections 3 and 4 we introduce the whole steps for the sequential and parallel implementation of the multipath method, respectively. In section 5 we show our results and finally we present our conclusions in section 6.

2 MULTIPATH METHOD

The multipath algorithm, described in (Sbert et al., 1996; Sbert, 1997), uses segments of global lines to build random walks that mimic classic random walks with infinite path length (see figure 1). The main differences as compared to the classic (local) random walk approach is the source probability selection which is proportional to the area of the patch, the simultaneous advance of different paths thanks to global lines, and transporting different logical paths in a single geometrical one. The first difference makes it only efficient in "smoothed" scenes, with emittance occupying a large part of the scene and more or less equilibrated. For this reason a *first shot* distributing direct illumination before applying the algorithm is necessary (Castro et al., 1998; Szirmay-Kalos et al., 2000). First shot method uses Monte Carlo local lines for the initial energy distribution from the light sources (see figure 2, right). After the first shot execution, it runs the multipath algorithm that uses Monte Carlo global lines to compute the light reflectances (see figure 2, left).

The multipath algorithm casts a predetermined number of random global lines using for instance, random points on an enclosing sphere. Each line will produce an intersection list, and the list is traversed taking into account each successive pair of patches (see figure 3). Each patch (if not emitter) stores two quantities. One records the power accumulated, and the other the unshot power. For every pair of patches along the intersection list, the first patch of the pair will transmit its unsent power to the second patch of the pair. So the unshot energy of the first patch is reset to zero, and the two quantities at the second patch, the accumulated and the unsent energy, are incremented. In the case of a source a third quantity i.e., the emitted power per line exiting the source is also kept. This power is precomputed in the following way: Given the number of lines to cast, the forecast number of lines passing through any source is found. This can be done with the Integral Geometry methods (Santaló, 1976). The division of the total source power by this number of forecast lines gives the predicted power of one line. Then, if the first patch of a pair

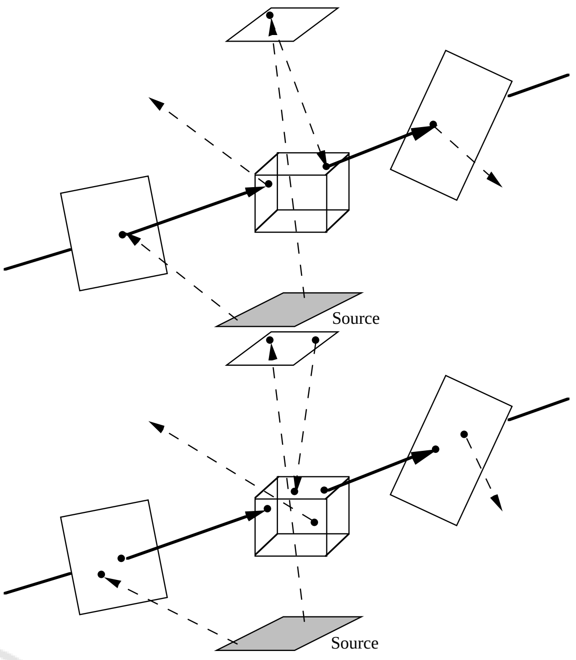

Figure 1: Random walk simulation with global lines. Top image, a global line (the thick continuous one) makes two paths advance at once (using previous global lines). Considering bidirectionality of the global lines, two other paths (cube surface and the more left polygon) will also advance in the reverse direction of the line. Down image, the exit point on each surface is random but the random walk simulation is still the same.

is a source patch, the power transported to the second patch of the pair will also include this predicted power portion. Considering bidirectionality, the same process is applied for the second patch of the pair of face to face patches.

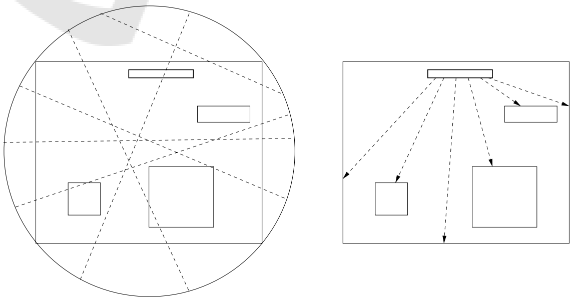

Figure 2: From left to right, global line and local line Monte Carlo methods. Global lines are cast from a bounding sphere and all intersections, that the line made with the scene, are considered. Local lines are cast from a patch and just the first intersection, that the line made with the scene, is considered.

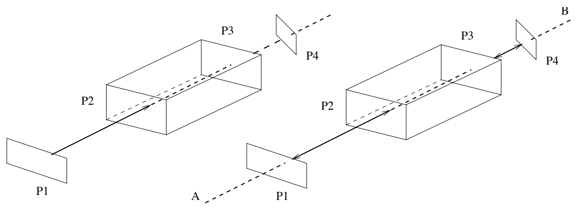

Figure 3: From left to right, the local line goes from the emitter patch to the first intersection of the line, there is a unidirectional energy sent from P1 to patch P2. On the right, the global line *AB* will transport power from patch P1 to P2 and from patch P3 to patch P4. There is no energy exchange between P2 and P3 because the related segment is inside an object. Considering bidirectionality, the line will also transport power from P2 to P1 and from P4 to P3.

3 SEQUENTIAL IMPLEMENTATION

The sequential implementation first reads the scene geometry and then the scene is subdivided in patches. Second, the first shot step, with a predefined, by the user, number of lines, is computed. Third, the multipath method, with predefined number of lines, is applied. Finally, the resulting scene is saved. This is summarized in figure 4. The first shot step and the multipath method are given by the following pseudocode:

```
begin firstShot()
   for i=0 to total number of patches do
     if patch[i] is a source then
       compute number of rays for this source
       according to its power
       for j=0 to number of rays do
         cast a local line    // see figure 2
         transfer energy from the source to
         the first intersected patch
       endFor
     endIf
   endFor
end

begin multipath()
   create a bounding sphere for the whole scene
   for i=0 to total number of rays do
     cast a global line       // see figure 2
     compute all intersections
     for all intersections do
       // see figure 3
       exchange power between pair of patches
     endFor
   endFor
end
```

It is easy to see that the more consuming parts, in our sequential implementation, are the `firstShot` and `multipath` steps. The idea is then to execute both functions in parallel.

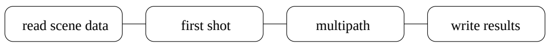

Figure 4: Sequential algorithm scheme of our radiosity implementation.

4 PARALLEL IMPLEMENTATION

In our parallel implementation we have two regions in our code: the sequential region and the parallel region. In our case the parallel region is given by the first shot step and the multipath method. Every cluster node has its own copy of the scene data and radiosity solution vector. Thus, every node computes its own solution of the first shot step and multipath method according to the code explained in previous section. The sequential region is given by three sections. The first section is loading the data at the beginning of the process. The second one is the combination of the results after the first shot and the distribution of this result to all the nodes. And the third one is the computation of the final results at the end of the process. In figure 5 we can see how all these tasks are structured in our implementation.

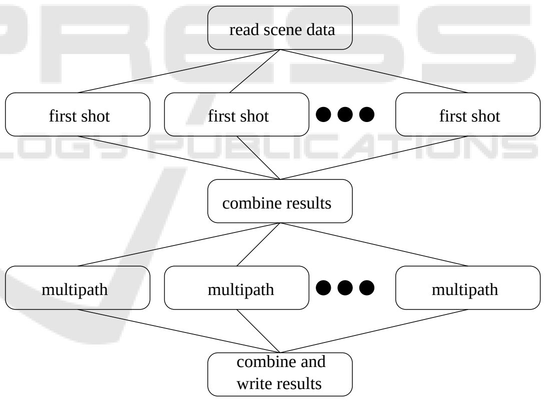

Figure 5: Parallel algorithm scheme of our radiosity implementation.

The tasks for each node is balanced because the predefined number of rays is divided by the number of processors. The cost of every line depends of the number of intersections. In the case of local lines the cost is more less the same but for global lines the cost can be very different. Considering that a scene needs millions of global lines then the average intersection cost will be almost the same at the end of the process.

On the other hand, the sequential cost depends of the communication cost plus the master node CPU time. Our communication cost is very low because there are just two communications steps in our implementation: one is bidirectional and the other one is unidirectional. The bidirectional communication process is used after the first shot step. The result of each node is sent to the master node and this node combines them. The final result is then sent back to each node in order to use this result as an initial values of the multipath method. The unidirectional communication step is applied at the end of the process. The task of combining the results is simple. The master node CPU has to add, for every scene patch, all the radiosity results and divide them by the number of processors. Thus the execution time, in the master node, is proportional to the number of patches.

5 RESULTS

We have used a high performance cluster composed by four servers HP ProLian DL145, each server has two AMD Opteron 244 (1.8GHz) processors and 2GB of RAM memory. Also a 100Mbps network switch is used. The cluster has been configured with OSCAR (Open Source Cluster Application Resources) on a GNU/Linux Suse 10.0 OS. We did our implementation in C++ using the SIR architecture (see (Martin et al., 1998)) and LAM/MPI (Message Passing Interface) libraries. We tested our implementation with the museum stairs and airplane cabin scenes (see figures 8 and 9).

Speed-up and Efficiency are measures that indicate how well a program has been parallelized. Let $T(p)$ be the execution time in p CPUs. The Speedup $S(p)$ and Efficiency $E(p)$ are defined as:

$$S(p) = \frac{T(1)}{T(p)} \quad p = 1,2,3...n$$

$$E(p) = \frac{S(p)}{p}$$

Figures 6 and 7 show the speed-up and efficiency, respectively, of our parallel implementation, for the museum stairs scene in figure 8. We can see that the speed-up keeps around 90% for different number of CPUs. Scalability shows that the efficiency $E(p)$ remains constant over a large number of processors. Following this idea, we can see that our implementation has a good scalability.

According to our experiments, in order to get a better result, the best proportion is 50% of local lines and 50% of global lines. But it is important to note that the bounding sphere volume is around 30% bigger than the bounding box volume of the scene. It means that 30% of the global lines will not intersect any scene polygon because are passing through an "empty" space.

The museum stairs scene was subdivided in 614.778 patches and 100 million rays for the first shot step and 130 million for the multipath method were cast. The execution time using the sequential implementation was 1346.4 seconds (484.7 seconds for the first shot and 861.7 for the multipath method) and for our parallel implementation, using 8 CPUs, was 189.5 seconds (70.2 seconds for the first shot and 119.3 for the multipath method).

The airplane cabin scene was subdivided in 438.518 patches and 100 million rays for the first shot step and 130 million for the multipath method were cast. The execution time for the sequential and parallel implementation was 951.3 (337.7 seconds for the first shot and 613.6 for the multipath method) and 130.5 seconds (47 seconds for the first shot and 83.5 for the multipath method), respectively. This scene shows similar speed-up and efficiency than the museum stairs scene.

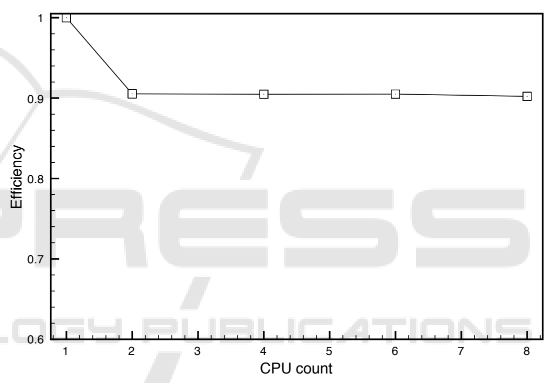

Figure 6: Efficiency for the museum stairs scene in figure 8.

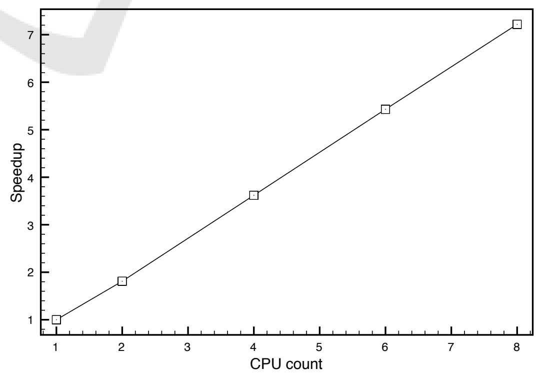

Figure 7: Efficiency for the museum stairs scene in figure 8.

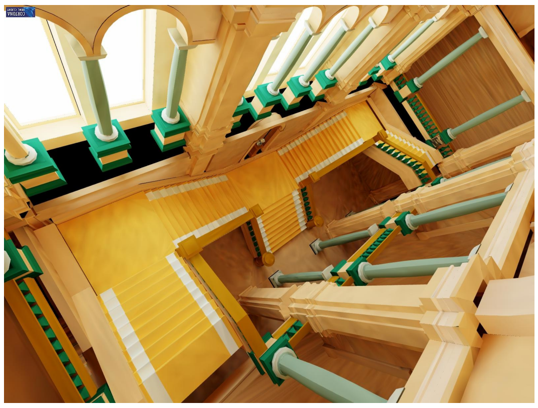
Figure 8: Museum stairs scene (a view from the top) with 614.778 patches computed using 100 million rays for the first shot step and 130 million for the multipath method.

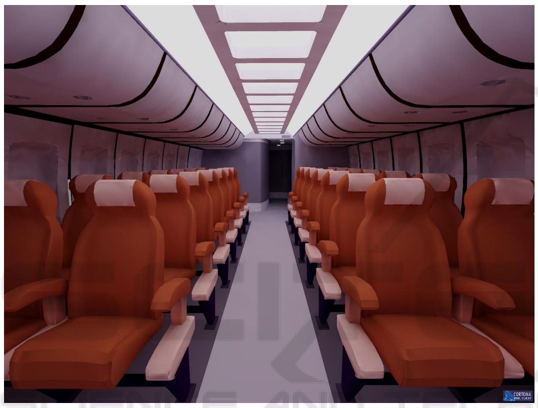
Figure 9: Airplane cabin scene with 438.518 patches computed using 100 million rays for the first shot step and 130 million for the multipath method.

6 CONCLUSIONS

We have presented a parallel implementation of the multipath method for radiosity. The implementation has been done in a cluster of PCs. Tests have been done for 2 to 8 processors, showing good efficiency and scalability.

As future work is possible to implement other related global line Monte Carlo algorithms. Also, other architectures, like multicore shared memory, are a good option for Monte Carlo methods.

ACKNOWLEDGEMENTS

This project has been funded in part with grant number TIN2007-68066-C04-01 of the Spanish Government.

REFERENCES

Alme, H., Rodrigue, G., and Zimmerman, G. (1998). Domain decomposition methods for parallel laser-tissue models with monte carlo transport. In Niederreiter, H. and Spanier, J., editors, *Proceedings of the Third International Conference on Monte Carlo and Quasi-Monte Carlo methods in Scientific Computing*, Claremont, California, USA. Springer-Verlag.

Bekaert, P. (1999). *Hierarchical and Stochastic Algorithms for Radiosity*. PhD thesis, Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium.

Besuievsky, G. and Pueyo, X. (1997). Making Global Monte Carlo Methods Useful: An Adaptive Approach for Radiosity. In *Actas VII Congreso Español de Informática Gráfica (CEIG '97)*, Barcelona, Spain.

Buckalew, C. and Fussell, D. (1989). Illumination Networks: Fast Realistic Rendering with General Reflectance Functions. In *Computer Graphics (ACM SIGGRAPH '89 Proceedings)*, volume 23, pages 89–98.

Castro, F., Martínez, R., and Sbert, M. (1998). Quasi-monte carlo and extended first-shot improvement to the multi-path method. In Szirmay-Kalos, L., editor, *Proc. Spring Conference on Computer Graphics '98*, pages 91–102, Budimerce, Slovakia. Comenius University. Available from http: //www.dcs.fmph.uniba.sk /˜sccg/ proceedings/ 1998.index.htm.

Martin, I., Perez, F., and Pueyo, X. (1998). The SIR Rendering Architecture. *Computers & Graphics*, 22(5):601–609.

Martínez, R. (2004). *Adaptive and Depth Buffer Solutions with Bundle of Parallel Rays for Global Line Monte Carlo Radiosity*. PhD thesis, Universitat Politècnica de Catalunya, Barcelona, Spain. Available from http://ima.udg.es/˜roel.

Reinhard, E., Chalmers, A. G., and Jansen, F. W. (1998). *Eurographics '98 State of the Art Reports*, chapter Overview of Parallel Photo-Realistic Graphics, pages 1–25. Available from http: // www.cs.bris.ac.uk /Tools /Reports /Authors / alan.html.

Santaló, L. A. (1976). *Integral Geometry and Geometric Probability*. Addison-Wesley, New York.

Sbert, M. (1997). *The Use of Global Random Directions to Compute Radiosity: Global Monte Carlo Techniques*. PhD thesis, Universitat Politècnica de Catalunya, Barcelona, Spain. Available from http://ima.udg.es/˜mateu.

Sbert, M., Perez, F., and Pueyo, X. (1995). Global Monte Carlo: A Progressive Solution. In Hanrahan, P. M. and Purgathofer, W., editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 231–239, New York, NY. Springer-Verlag.

Sbert, M., Pueyo, X., Neumann, L., and Purgathofer, W. (1996). Global Multipath Monte Carlo Algorithms for Radiosity. *The Visual Computer*, 12(2):47–61.

Szirmay-Kalos, L. (1999). Stochastic Iteration for Non-Diffuse Global Illumination. In *Computer Graphics Forum (Proceedings Eurographics '99)*, volume 18, pages C–233–C–244.

Szirmay-Kalos, L. and Purgathofer, W. (1998). Global Ray-Bundle Tracing with Hardware Acceleration. In Drettakis, G. and Max, N., editors, *Rendering Techniques '98 (Proceedings of Eurographics Rendering Workshop '98)*, pages 247–258, New York, NY. Springer Wien.

Szirmay-Kalos, L., Sbert, M., Martínez, R., and Tobler, R. F. (2000). Incoming First-Shot for Non-Diffuse Global Illumination. In *Spring Conference on Computer Graphics*, Budmerice, Slovakia. Available from http://www.fsz.bme.hu/˜szirmay/puba.htm.

Zareski, D., Wade, B., Hubbard, P., and Shirley, P. (1995). Efficient parallel global illumination using density estimation. In *Proceedings of Visualization '95 - Parallel Rendering Symposium*, pages 219–230.