Speculative parallelization of multipath radiosity algorithm

A. Trias, [1] J. Puiggalí,[1] F. Castro,[1] T. Jové,[1], M. Sbert[1] and J. L. Marzo[1]
[1] Institut d'Informàtica i Aplicacions (IIiA)
Universitat de Girona, 17071 Girona, Spain
{albert.trias,joan.puiggali,francesc.castro,teodor.jove,mateu.sbert,joseluis.marzo}@udg.edu

***Abstract*—In computer graphics, global illumination algorithms take into account not only the light that comes directly from the sources, but also the light interreflections. This kind of algorithms produce very realistic images, but at a high computational cost, especially when dealing with complex environments. Parallel computation has been successfully applied to such algorithms in order to make it possible to compute highly-realistic images in a reasonable time. We introduce here a speculation-based parallel solution for a global illumination algorithm in the context of radiosity, in which we have taken advantage of the hierarchical nature of such an algorithm.**

***Index Terms*—global illumination, radiosity, distributed environments, speculation, parallelization, computer clusters.**

I. Introduction

Radiosity techniques [1], [2] aim at estimating the illumination for environments with diffuse reflecting surfaces in order to produce highly-realistic results. The multipath algorithm [3] is a Monte Carlo technique that solves the radiosity problem by using random lines which simulate the exchange of radiant power among the objects in the environment.

In the present paper we introduce the parallelization of a variant [4] of the multipath algorithm in which the environment is structured in a hierarchy of subscenes, allowing the algorithm to be run at different levels, which involves a better exploitation of the random lines. We aim at taking advantage of the parallelism at the maximum, in order to accelerate the computations.

The hierarchical nature of the algorithm to be parallelized has allowed us to apply a master/slave speculative parallelization architecture for computer clusters. A noticeable parallelization efficiency has been obtained in our experiments.

The paper is organized as follows. In section II we present the previous work. Section III describes in detail the contributions presented in this article, namely the parallel strategy we have used. Section IV presents the results obtained in our experiments. Finally, section V summarizes the conclusions of the paper and presents some future lines of research.

II. Previous work

A. Radiosity

Radiosity [1], [2] is a global illumination algorithm that simulates the multiple reflections of light around a scene. The radiosity model assumes that the exiting radiance of a point is direction independent, which is valid to simulate diffuse reflections. The equation for the radiosity $B(x)$ (emitted plus reflected light leaving point x) is:

$$B(x) = \int_D \rho(x)F(x,y)B(y)dy + E(x), \tag{1}$$

where D stands for the set of all the surfaces in the scene, $\rho(x)$ and $E(x)$ respectively stand for the *reflectance* (fraction of the incoming light that is reflected) and *emittance* (emission radiosity if x is a light source) at point x, and $F(x,y)$ stands for the *form-factor* between points x and y, a geometric term which includes the visibility between x and y.

In practice, a discrete version of the radiosity equation is considered, corresponding to a discretization of the environment in small polygons called *patches*:

$$B_i = E_i + \rho_i \sum_{j=1}^{N_p} F_{ij}B_j, \tag{2}$$

where B_i, E_i, and ρ_i stand respectively for the radiosity, emittance, and reflectance of patch i, N_p stands for the number of patches in the scene, and F_{ij} stands for the patch-to-patch form factor from patch i to patch j.

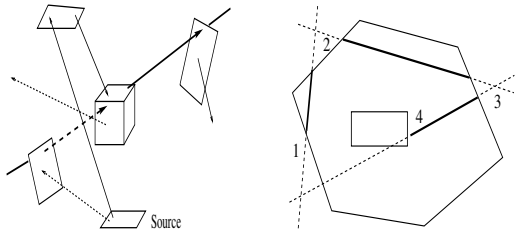
B. The multipath algorithm

The radiosity multipath algorithm was first described in [3]. It is a member of a family of methods called by different authors global Monte Carlo, global radiosity, or transillumination methods [5], [6], [7] which use random global lines (or directions) to transport energy. Global lines are independent of the surfaces or patches in the scene, contrary to local lines, and can take advantage of all their intersections with the scene.

The multipath algorithm can be seen as a random walk whose transition probabilities are given by the form factors. Such a random walk is generated from a uniform density of global lines [8]. Each global line simulates the exchange of energy between several pairs of patches, contributing to several geometric paths (Fig. 1a).

Let us briefly review the algorithm. Global lines are sequentially intersected with the scene. For each line, intersections are sorted by distance, resulting in a list of pairs of intersected patches. Each patch keeps two amounts: its *accumulated* outgoing power, and its *unshot* outgoing power. The patches in each pair along the intersection list exchange their unshot

power times the corresponding reflectances, and sum this amount to their corresponding accumulated power. If a patch is a source, we also keep a third quantity, the *emission power per line* exiting the source. Thus, if one of the patches of the pair is a source, we must add, at each exchange, its emission power per line to its unshot power. This "emitted per line" power is previously computed by dividing the total emission power of the patch by the forecast number of lines crossing the patch, which is proportional to the area of the patch [8].



(a) (b)

Fig. 1. Multipath algorithm. (a) A global line (the thick one) simulates two paths, indicated with the continuous stroke and the dashed stroke (b) A path can contribute to the emission of radiant power from several patches. In the figure, path 1-2-3-4 simulates logical paths 1-2-3-4, 2-3-4 and 3-4.

C. The multipath algorithm using a hierarchy of subscenes

A variant of the multipath algorithm was presented in [4], where a hierarchy of subscenes was used in order to run multipath not only for the whole scene but also for the subscenes, submitting each subscene to its own density of lines (referred to as *locally global lines*). This arised from the idea of better exploiting the lines by using more lines where they were more necessary. We have to note that such densities of lines were uniform in the context of each subscene but not in the context of the whole scene. Next we elaborate on the main features of this algorithm:

1) Hierarchy of subscenes and adaptive densities of lines: The hierarchy of subscenes is built using a clustering bottom-up strategy [9]. Fig. 2 shows a simple example of a 2-level hierarchy, although the algorithm allows any number of levels. Note in the example that a subscene can contain other subscenes and/or single objects.

2) Subdivision of the bounding boxes: virtual patches and angular regions: Each face of each bounding box (unless the box at the top level) acts as a *virtual wall* (VW), which is subdivided in a grid of *virtual patches* (VP), as seen in Fig. 3 (left). The directional hemisphere over each virtual patch is subdivided in *angular regions* (AR) (see Fig. 3 (right)). Each AR acts as an accumulator of undistributed incoming and outgoing power.

3) The pre-process and the reuse of lines: Each subscene is submitted into a uniform density [8] of locally global lines. For each line cast in a subscene S, a sorted-by-distance intersection list is computed, considering the intersections of the line against:

- Single objects inside S.

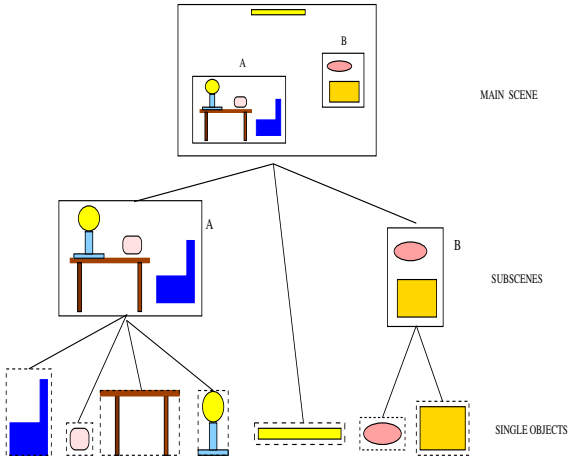
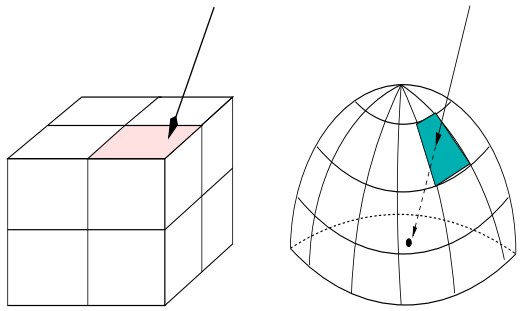


Fig. 2. A 2-level hierarchy of subscenes.



(a) (b)

Fig. 3. (a) Subdivision of the bounding box virtual wall in a grid of virtual patches. (b) Directional subdivision in angular regions over a virtual patch.

- Virtual walls of the inner subscenes of S (ignoring their interior).
- Virtual walls of S.

This *pre-process* allows to obtain information about the geometry of the subscenes, estimating the *transmittance* for each AR. The transmittance T_R for the angular region R represents the fraction of power entering subscene S in the directions in R that crosses S finding no obstacle in its way. Transmittances give an idea about the opaqueness of a subscene in each direction, allowing, with a moderate loss of accuracy, to ignore the interior of the subscenes when executing multipath at a higher level. The transmittance T_R is estimated as

$$T_R \approx \frac{n_R^d}{n_R}, \tag{3}$$

where n_R^d is the number of lines crossing S in the directions in R and hitting no object, and n_R is the total number of lines crossing S in the directions in R. If $n_R = 0$, T_R is estimated by interpolation of the values of the neighboring regions.

For each line, and after sorting by distance, the identifiers of the intersected patches and angular regions are stored. This

information is repeatedly used in the iterative process, avoiding to compute again the intersections.

4) The algorithm: The full algorithm is presented in Fig. 4. Note that recursive function MP (Fig. 5) deals with the exchange of power inside and between each subscene. Note also that a first-shot stage [10] is needed before applying multipath, in order to efficiently distribute the direct illumination. Thus, multipath only estimates the indirect illumination.

```
Generate hierarchy of subscenes
Subdivide each VW of the boxes in VP and AR
for each subscene S (including the whole scene)
    Cast global lines and store intersection lists
    if S is not the whole scene
      for each AR in S
          Compute and store its transmittance
      endfor
    endif
endfor
First shot (computing direct illumination)
for each iteration (4 or 5 are usually enough)
       MP(whole scene) // RECURSIVE FUNCTION
endfor
```

Fig. 4. The iterative algorithm.

D. Master/Slave Speculative Parallelization Architecture for Computer Clusters (MSSPACC)

Master/Slave Speculative Parallelization Architecture for Computer Clusters [11], [12], [13] achieves parallelism by using speculation in distributed environments, allowing the parellel execution of a sequential program in a computer cluster. It simulates the behavior of a superscalar system by implementing a form of parallelism called instruction-level parallelism. These techniques try to break true data dependencies and control dependencies by means of using speculation of future data values and future branch results respectively. Speculation is based on the fact that the program behaviour is usually repetitive and consequently predictable, as demonstrated in studies of branch [14], memory dependencies, and data values [15]. These techniques can be classified into two types:

- Software speculations: Compilers carry out the necessary coding. It cannot be applied dynamically [16], [17], [18].
- Hardware speculations: They require duplicated hardware elements, e.g. adding extra registers to store provisional values until they are definitive [15], [19], [20].

The above techniques allow the processor to divide program execution into several thread executions, and increase the program's degree of parallelism. Moore's law (processing power doubles each 18 months) and Gilder's law (bandwidth triples each 12 months) show that the costs of information transmission and synchronization between workstations decrease faster than processing speed. These premises make it

```
procedure MP(scene S)
Compute power per line for each patch in S
not included in any subscene of S
for each AR in S
    Compute incoming power per line
    Set to 0 accumulated outgoing power
endfor
for each subscene S_i of S
    for each AR in the box of S_i
        Compute outgoing power per line
        Set to 0 accumulated incoming power per line
    endfor
endfor
for each line cast in S
    Faced patches and/or AR exchange power:
        * Patches contribute unshot power + power per line
        * AR of S contribute incoming power per line
        * AR of S_1...S_k contribute
          outgoing power per line
        * Power leaving S is accumulated as outgoing
          power in the corresponding AR
        * Power crossing S_1..S_k is accumulated as incoming
          power in the corresponding AR and attenuated
          by the corresponding transmittance
endfor
for each subscene S_i of S
    MP(S_i)                  // RECURSIVE
endfor
endprocedure
```

Fig. 5. The recursive function MP. $S_1..S_k$ are the subscenes of S.

possible the idea of transporting speculation techniques to a distributed environment composed by cheap workstations. The complete design of Master/Slave Speculative Parallelization Architecture for Computer Clusters system [11], [12], [13] consists of three subsystems:

- The parallelizing subsystem (Fig.6) transforms the original sequential program into the parallel format needed by the execution environment. The program is divided into blocks that can be executed in parallel. Either two or three programs (depending on the type of the original program) are generated as a result of the translation process: a master, a slave, and optionally a master/slave. The master manages the parallelism and the speculation of the system. The slave runs, at each of the nodes, the code of the blocks into which the sequential program has been divided. The master/slave program permits to reduce the master bottleneck by distributing the tasks to others nodes that work as a new master.
- The execution subsystem (see Fig.7) runs, by applying speculation, the parallelized applications in a computer cluster composed by a monoprocessor architecture using PVM (Parallel Virtual Machine). The execution environ-

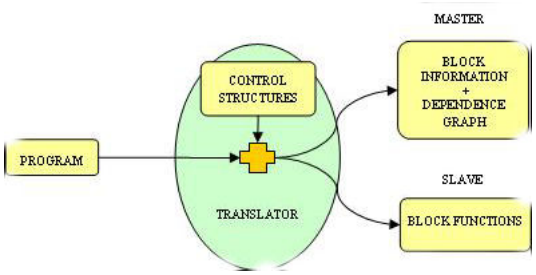


Fig. 6. The parallelizing subsystem

ment behaves like a superscalar processor, where the blocks are like the instructions into which the sequential program has been divided, and the nodes where the slave program is run are like the functional units. There exist the following data speculation mechanisms: Data Value Speculation [21]; Last Value Predictor [21]; Stride Predictor [22]; and Context-based Value Predictor [23]. Control dependencies are managed with branch prediction techniques based on a BTB (Branch Target Buffer) with 2-bit history [24]. If there are blocks executed with wrong speculated values or branch missprediction, such blocks are discarded and their execution is restarted from the last stable point.

- The simulation subsystem (see Fig.7) is developed in order to evaluate the impact of the technological evolution or the effect of using bigger computer clusters. The simulation is able to be run in a single workstation, using the information obtained from the monoprocessor execution (the trace of the program), and the cluster execution (the execution cost of the different blocks).

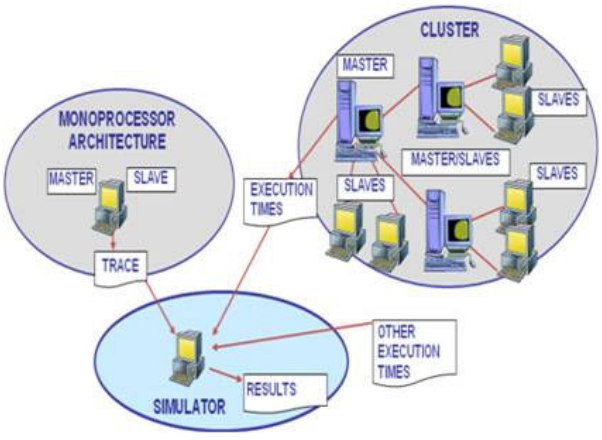


Fig. 7. Execution and Simulation Subsystem

III. Our contribution

To apply the speculation techniques seen in section II-D to the multipath algorithm (section II-C) it has been needed to avoid the use of dynamic memory, because the cluster's nodes do not use shared memory, and the objects positions depend on the node. We have also avoided recursivity because it produces data dependencies which limit parallel execution. We have tried to obtain a code as sequential as possible to get more flexibility defining the blocks. An example of a block where these transformations have been done (recursivity and dynamic memory have been removed) is shown in Fig. 9, while in Fig. 8 we have the orignial methods.

```
procedure DividePolygons()
for each polygon i
    polygon[i].divide()
endfor
endprocedure

procedure Polygon::divide()
if area() > maximum area allowed
    for j = 0 to 3
        addLeaf(division(j))
        getLeaf(j).divide()
    endfor
endif
endprocedure
```

Fig. 8. Polygons subdivision

```
block Divide Polygons
N = amount of polygons to divide
i = 0
while i < N
    if polygon[i].area() > maximum area allowed
        polygon[i].setLeafPointer(N)
        for j = 0 to 3
            polygon[N] = polygon[i].division(j)
            N = N + 1
        endfor
    endif
    i = i + 1
endwhile
endblock
```

Fig. 9. Removing recursivity and dynamic memory from methods in Fig. 8

A. Basic blocks

The execution subsystem needs to divide in basic blocks the algorithm to be executed. In our case, such a division can be seen in Fig. 10, where:

- *Parallelizable loop* means that the iterations can be executed at once using speculation on the induction variable.
- *Non-parallelizable loop* means that there are dependencies that cannot be solved by speculation.

B. Algorithm modifications

In a given iteration (see algorithm in Fig. 4), the dependencies because of the power accumulators in the AR bear that

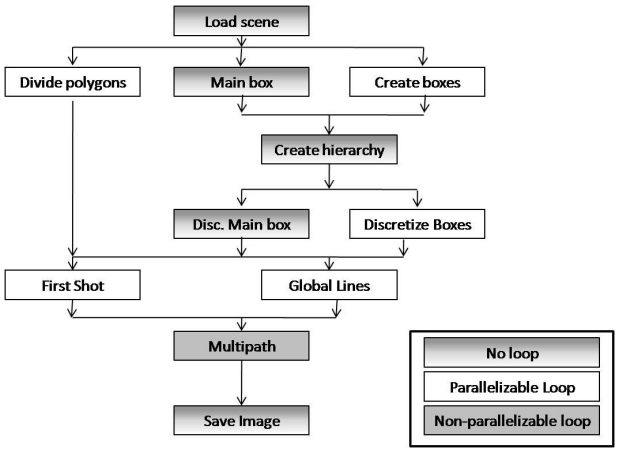


Fig. 10. Block division for the master node.

we can only run at once MP for the boxes of the same level. If power accumulated in the AR was not used until the next iteration, MP could be run at once for all the boxes. Thus, we decide to duplicate the AR accumulators of incoming and outgoing power, keeping the power accumulated in the current iteration and distributing it in the following one.

For a given execution of multipath, increasing the performance of parallel execution involves breaking dependencies on the unshot power variables (see section II-B) between the global lines. In this sense, we add a new field for each patch that accumulates the unshot power of a whole multipath iteration in order to be distributed in the following one. One of the problems of such a modification is that the unshot power of the last iteration will be never distributed. Such a problem has been reduced by increasing the number of iterations, which reduces the amount of unresolved power after the last iteration.

Since the MSSPAC has a limitation on the amount of embedded loops that can be managed, and it is not possible to run multipath iterations at once, we applied a loop-unrolling in the Multipath block, so that it is possible to run at once all the boxes and use several slaves for each box. We also reduced the hard disk operations by increasing the buffer size.

IV. Results

In our experiments we have employed the scene office, which represents a room with 4 tables, a desk and five chairs. The light source is a lamp stuck on the center of the ceiling. Our experiments have been done in a cluster of Pentium IV with 3 Ghz and 1 Gbyte of RAM. The PCs, which are connected in a LAN, have a shared hard disk, where the global lines have been stored.

As shown in Fig. 11, the use of the speculation techniques has reduced the cost with respect to the sequential implementation of the algorithm. Fig. 11 also shows the tendency of the cost to stabilize from a given number of nodes on. The speed-up of the parallel implementation regarding to the number of nodes can be seen in Fig. 12, while the relative efficiency, that is, the speed-up divided by the number of nodes, is shown in Fig. 13, where it can be seen that the relative efficiency is optimal near approximately 20 nodes, tending to decrease from this value on.

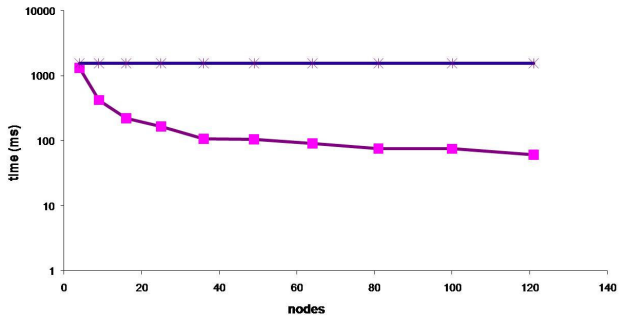


Fig. 11. Running time vs. number of nodes. Horizontal line shows the running time for the sequential implementation when generating the same number of lines.

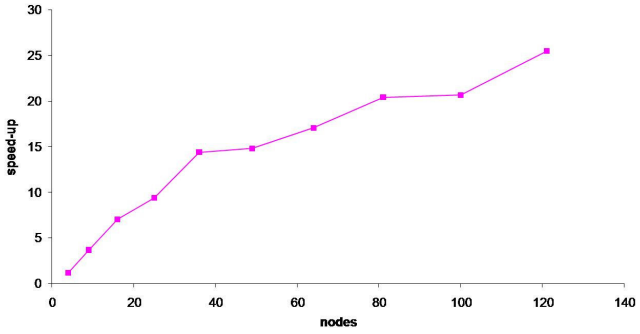


Fig. 12. Speed-up of our parallel implementation regarding to the number of nodes.

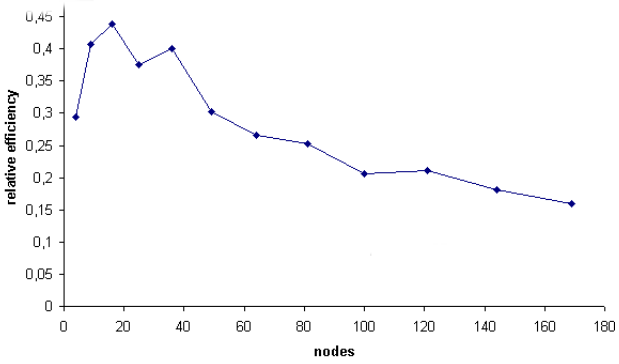


Fig. 13. Relative efficiency of our parallel implementation regarding to the number of nodes.

It is worth to be mentioned that, since speculation allows to obtain a higher degree of parallelization, the speed-up obtained with a parallel implementation without speculation would be lower than the one presented above. On the other hand, we have to remark that a slight bias has been introduced by the speculative parallel implementation, due to the undistributed

power after the last iteration (see section III-B). Such a bias is visually imperceptible, however, as we can see in the images in Fig. 14 and 15, in which we show some details of the scene office from a radiosity solution obtained with our parallel implementation.

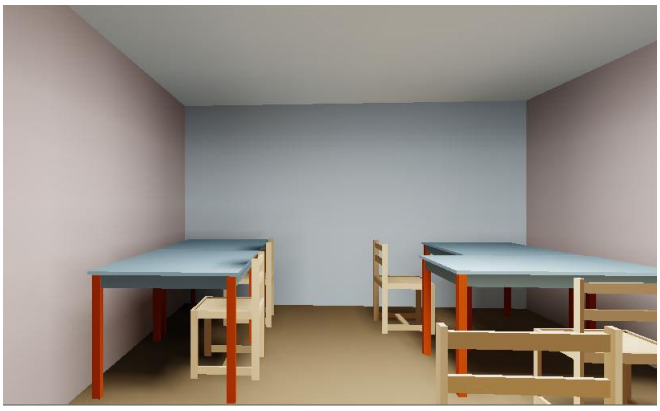


Fig. 14. Detail of scene office

Fig. 15. Detail of scene office

V. Conclusions and future work

The main goal of this work has been to improve the performance of the multipath algorithm, a Monte Carlo radiosity method that simulates the illumination of an environment. Taking advantage of a space-division-based hierarchical version of multipath, it has been parallelized using speculation in a distributed Master/Slave Speculative Parallelization Architecture for Computer Clusters [11], [12], [13].

The obtained results show that such a goal has been achieved by applying data and control speculation. These techniques try to use the maximum number of nodes in parallel by increasing the inherent degree of parallelism of the algorithm, and thus obtaining better performance from the processors. We conclude that such a degree of parallelism depends on many factors, such as scene complexity, space division, direct lighting distribution, and number of Monte Carlo samples. One of the first tasks carried out was to obtain a mathematical model that represents the behavior of the execution of the algorithm. Such a model has allowed us to carry out empirical studies on the parallel implementation, for example on the selection of the size of the blocks (pieces in which we divide the program).

Our implementation of the algorithm in the engine of speculation (MSSPACC) has allowed us to extract conclusions, but also new ideas in order to improve the performance by meeting some engine lacks. We have also proposed the implementation of the algorithm in a multiprocessor to compare an environment where the communication between the processors is based on the pass of messages (clusters) to an environment based on shared memory and the PVM (multiprocessor). This comparison will allow us to know the advantages and drawbacks of both models and to explore the possibility of mixing both models according to the problem evolution.

Acknowledgments

This project has been funded in part with grants number TIN2007-68066-C04-01, TIN2008-06862-C04-02/TSI, and TEC2006-03883/TCM from the Spanish Government, partially supported by the European Union project N 216746 and Universitat de Girona research grant BR09/10 awarded to Albert Trias.

References

[1] M. Cohen and J. Wallace, *Radiosity and Realistic Image Synthesis*. Academic Press Professional, 1993.

[2] F. Sillion and C. Puech, *Radiosity and Global Illumination*. Morgan Kaufmann Pubilishers, Inc., 1994.

[3] M. Sbert, X. Pueyo, L. Neumann, and W. Purgathofer, "Global multipath monte carlo algorithms for radiosity," *The Visual Computer*, pp. 47–61, 1996.

[4] F. Castro, M. Sbert, and L. Neumann, "Fast multipath radiosity using hierarchical subscenes," *Computer Graphics Forum, vol.23(1),*, pp. 43–53, 2004.

[5] M. Sbert, F. Pérez, and X. Pueyo, "Global monte carlo. a progressive solution," *Rendering Techniques '95. Springer Wien New York*, pp. 231–239, 1995.

[6] L. Neumann, "Monte carlo radiosity," *Computing, 55*, pp. 23–42, 1995.

[7] L. Szirmay-Kalos, T. Foris, L. Neumann, and B. Csebfalvi, "An analysis of quasi-monte-carlo integration applied to the transillumination radiosity method," *Proceedings of Eurographics 97*, 1997.

[8] L. Santaló, *Integral Geometry and Geometric Probability*. Ed. Addison-Wesley, New York, 1976.

[9] J. Goldsmith and J. Salmon, "Automatic creation of object hierarchies for ray tracing," *IEEE Computer Graphics and Applications*, vol. 7, no. 5, pp. 14–20, 1987.

[10] F. Castro, R. Martinez, and M. Sbert, "Quasi-monte-carlo and extended first shot improvements to the multipath method," *Proceedings SCCG '98, (Budmerice, Slovakia)*, 1998.

[11] J. Puiggalí, T. Jové, S. Salanova, and J. Marzo, "Execution speed up using speculation techniques in computer clusters." *International Mediterranean Modelling Multiconference (IMM 06)*, pp. 561–568, 2006.

[12] ——, "Execution limits of tls execution of sequencial programs on clusters." *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 06)*, pp. 12–19, 2006.

[13] J. Puiggalí, T. Jové, J. Segovia, and J. Marzo, "Master/slave speculative parallelization architecture for computer clusters." *ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2007)*, 2007.

[14] J. E. Smith, "A study of branch prediction strategies." *In Proceedings of the 8th Annual Symposium on Computer Architecture.*, pp. 135–148, 1981.

[15] C. B., G. Reinman, and D. Tullsen, "Selective value prediction." *In Proceedings of the 26th Annual International Symposium on Computer Architecture.*, 1999.

[16] C. Zilles and G. Sohi, "Master/slave speculative parallelization," *Proc. of the 35th Int. Symp. on Microarchitecture.*, 2002.
[17] J. Steffan, C. Colohan, A. Zhain, and T. Mowry, "Improving value communication for thread-level speculation." *Intl. Symp. on High-Performance Computer Architecture*, 2002.
[18] E. Larson and T. Austin, "Compiler controlled value prediction using branch predictor based confidence." *In Proceedings of the 33nd Annual ACM/IEEE International Symposium on Microarchitecture.*, 2000.
[19] J. Gonzalez and A. Gonzalez, "The potencial of data value speculation to boosst ilp." *Proceedings of the 12th International Conference of Supercomputing.*, 1998.
[20] Q. Jacobson, S. Bennett, N. Sharma, and J. Smith, "Control flow speculation in multiscalar processors." *In To Appear in Proceedings of the Third International Symposium on High-Performance Computer Architecture.*, 1997.
[21] M. H. Lipasti, W. C. B., and J. P. Shen, "Value locality and data speculation." *in Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems.*, pp. 138–147, 1996.
[22] Y. Sazeides, S. Vassiliadis, and J. Smith, "The performance potential of data dependence speculation and collapsing," *9th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-29)*, 1996.
[23] Y. Sazeides and J. Smith, "The predictability of data values," *10th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 1997.
[24] T. Yeh and Y. Patt, "Two-level adaptive branch prediction." *Proceedings of the 24th ACM/IEEE International Symposium on Microarchitecture*, pp. 51–61, 1991.