Fast and Accurate Wavelet Radiosity Computations Using High-End Platforms

Laurent Alonso, Xavier Cavin, Jean-Christophe Ulysse, Jean-Claude Paul

Fast and Accurate Wavelet Radiosity Computations Using High–End Platforms

Laurent Alonso[1], Xavier Cavin[2], Jean–Christophe Ulysse[2] and Jean–Claude Paul[1]
ISA research team
LORIA[3]
Campus Scientifique, BP 239
F–54506 Vandœuvre-lès-Nancy CEDEX, France
Xavier.Cavin@loria.fr

Abstract. In this paper, we show how to fully exploit the capabilities of high–end SGI graphics and parallel machines to perform radiosity computations on scenes made of complex shapes both quickly and accurately. Overlapping multi–processing and multi–pipeline graphics accelerations on one hand, and incorporating recent research works on wavelet radiosity on the other hand, allows radiosity to become a practical tool for interactive design.

1 Introduction

Since radiosity methods are becoming more frequently considered for a new wide range of applications (design, computer animations, virtual reality), efficient and accurate algorithms have to be found to provide a direct rendering of scenes made of complex shapes. Scenes created with modeling software typically include parametric surfaces — such as NURBS, cylinders, spheres —, arbitrary planar primitives, and their CSG combinations. Figures 1, 4, 5 and 7 illustrate several examples of such scenes, modeled in different representations.

Up–to–now, many techniques have been proposed to solve the famous "radiosity equation" for such surfaces, but unfortunately none of them appear to have found the right compromise between speed and accuracy. For instance, the clustering approach [14, 22, 24] allows very fast computations, but of a limited accuracy. On the contrary, wavelet radiosity [15], which is a generalization of the hierarchical radiosity method [16], provides a better approximation of the radiosity function, but at a non negligible computational cost. Attempts to parallelize hierarchical radiosity have been undertaken [9], but they have mostly proven the difficulty of this task.

In this paper, we fully exploit the capabilities of high–end SGI graphics and parallel machines to illuminate scenes made of complex shapes both quickly and accurately (see fig. 1), two goals that were considered incompatible with previous methods. Moreover, thanks to the incorporation of recent research works on wavelet radiosity that allow complex surfaces to be illuminated as if they were simple primitives [1, 12, 18], our parallel algorithm results in a better approximation of the radiosity function with a faster convergence and lower memory costs.

[1] INRIA Lorraine.

[2] Institut National Polytechnique de Lorraine.

[3] UMR n° 7503 LORIA, a joint research laboratory between CNRS, Institut National Polytechnique de Lorraine, INRIA, Université Henri Poincaré and Université Nancy 2.

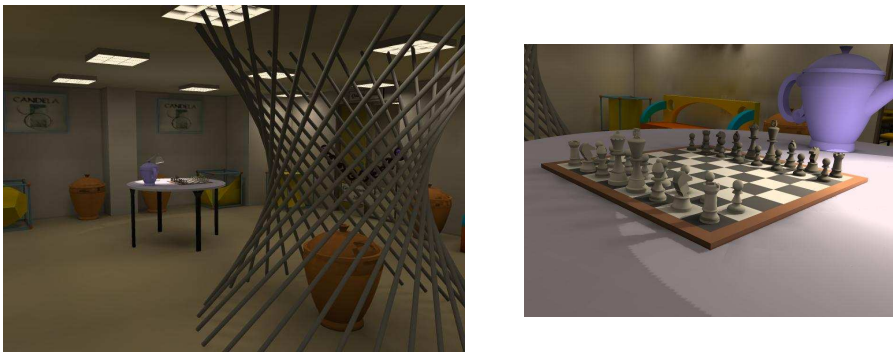


Fig. 1. The virtual geometry room in the 4th floor of the Soda Hall has been illuminated with our high–order wavelet parallel radiosity algorithm using recent state–of–the–art radiosity techniques. Rendering this complex scene —modeled with quadric surface patches (including the chess set) —with a similar speed and accuracy would have been impossible with previous radiosity methods, since the number of initial patches would have been intractable. Modeled in SGDL. By courtesy of SGDL Systems.

Our paper is organized as follows: in section 2, we review previous work on related radiosity techniques. Section 3 details the latest improvements of our parallel wavelet radiosity algorithm [7, 8] that have led to its current implementation. Section 4 is devoted to the experiments we have conducted with our application on different test scenes, including a full case study with the virtual SGI Reality Center™. Finally, in section 5, we draw some conclusions.

2 Previous work

Radiosity techniques have interesting properties that make them useful for new applications that require highly realistic virtual environments. These cover architectural and industrial design, but also the entertainment industry (as shown by Siggraph'99 panel: "Get Real! Global Illumination for Film, Broadcast, and Game Production"). Then, radiosity algorithms have to provide efficient and accurate solutions for the kind of scenes these potential users generally manipulate.

Scenes constructed using CAD tools — such as CATIA® or ARC+™ — or more general purpose modelers — such as Maya™, 3D Studio MAX® or SGDL — frequently contain complex planar surfaces, with curved boundaries or holes in them (see fig. 4(a)), and/or curved and parametric surfaces (see fig. 1 and fig. 7). Moreover, in the future, surfaces extracted by 3D scanning systems will routinely be transformed into a set of parametric surfaces.

We analyze in section 2.1 the two main general approaches that would traditionally be applied: clustering and wavelet radiosity. Since none of them appear to find the right compromise between speed and accuracy, section 2.2 briefly introduces recent research advances on which the remaining of this paper will be based.

2.1 Classical approaches

Probably the most widely used approach to deal with complex surfaces in radiosity is to tessellate them into simple planar surfaces (only little research has been devoted to

arbitrary planar polygons [3, 4] and parametric surfaces [10, 21, 25, 29]). This method has several negative consequences on the radiosity algorithm:

- It increases the number of input surfaces the algorithm must handle.
- The tessellation is made as an artificial pre–treatment before the radiosity computations, influencing these computations and creating arbitrary discontinuities. It can prevent us from reaching a good illumination solution.
- It does not allow a hierarchical approach for the radiosity function on the original surface, much less higher order wavelets.
- The tessellation can create poorly shaped triangles (see fig. 4(b)), which can cause Z–buffer artifacts at the visualization stage, and are harder to detect in visibility tests.

Clustering. Obviously, some of the tessellation drawbacks, such as the number of initial surfaces, can be addressed using clustering [14, 22, 24]. In clustering, neighboring patches are grouped together, into a *cluster*. The cluster receives radiosity and dispatches it to the patches it contains. Unfortunately, increasing the number of initial surfaces — for a better approximation of original surfaces — makes it harder for the clustering approaches to compute an optimal cluster hierarchy [17]. Also, tessellating the original surface, then clustering the pieces has removed the connection to the original surface, resulting in numerous approximations. It would probably be more efficient to apply clustering to the original surfaces rather than to the result of their tessellation.

A better grouping strategy is face–clustering [27]. In face–clustering, neighboring patches are grouped together according to their coplanarity. Yet, even face–clustering depends on the geometry created by the tessellation, and it does not remove the problems due to the tessellation approximations, such as the calculation of the exact spatial position of the points or their normals.

Wavelet radiosity. Clustering methods can be considered when one wants to obtain a fast, but approximate, rendering of radiosity simulations. When physical accuracy is the objective, one of the most viable alternative is wavelet radiosity. The wavelet radiosity method was introduced by [15]. It is an extension of the hierarchical radiosity method [16] that allows the use of higher order basis functions.

In theory, higher order wavelets are providing a more compact representation of complex functions. Hence, they use less memory and give a smoother representation of the radiosity function, that looks better on display. However, in a previous experimental study [26], the practical problems of higher order wavelets were largely negating their theoretical benefits, thus prohibiting any rapid computation.

Moreover, the algorithmic complexity of the hierarchical radiosity algorithm is quadratic with respect to the number of initial surfaces, making it not particularly well adapted to highly tessellated surfaces.

The parallelization of the hierarchical radiosity algorithm has been explored in several studies [7, 8, 9, 13, 20, 23] in order to overcome the limitations due to its high computational cost. All the authors have converged to the conclusion that its non–uniform, dynamically changing characteristics, and its need for long–range communication made it less suitable for effective parallelization.

2.2 A novel approach

Since clustering methods allow fast but approximate computations, while wavelet radiosity algorithms, even the parallel ones, are more precise but slower, should we give up and conclude that radiosity is either a solved problem or an insoluble one?

Actually, recent innovative research works on wavelet radiosity have shown that highly accurate simulations could be performed using higher order wavelets [12], both on arbitrary planar surfaces, thanks to the *Extended Domain Algorithm* [18], and on parametric surfaces, using *Constant Jacobian Mappings* [1]. These new techniques completely replace the need for tessellating the original surfaces. By exactly integrating these surfaces as a single entity in the resolution process — thus reducing the algorithmic complexity of the wavelet radiosity algorithm —, complex scenes can be rendered more quickly, more accurately and much more naturally than with previously known methods.

We present in the remainder of this paper how we can advantageously integrate all these promising advances to build a new parallel algorithm, based on our previous work on parallel wavelet radiosity [7] and combined hardware accelerated visibility [8]. Running a state–of–the–art wavelet radiosity algorithm that intensively uses all the parallel and graphics resources of high–end SGI machines, we show that fast rendering of accurate radiosity simulations of complex scenes has finally become a reality.

3 Latest improvements of parallel wavelet radiosity

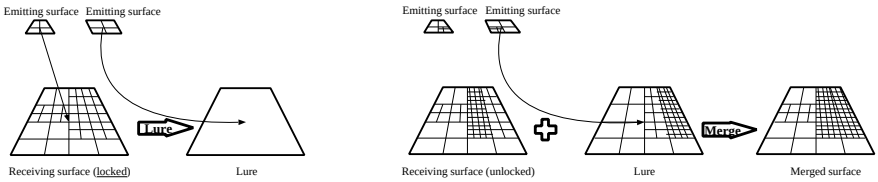
In this section, we present our new parallel algorithm for fast and accurate wavelet radiosity computations on complex surfaces. Our algorithm is the natural extension of previous work on parallel and hardware accelerated radiosity [7, 8] to the most recent research on wavelet radiosity [1, 12, 18]. Here, the original input surfaces are no longer tessellated into a set of planar surfaces, but directly treated as a single entity by the resolution process.

Section 3.1 explains how the radiosity computations between the complex input surfaces are performed in parallel. Then, section 3.2 deals with how to perform hardware based visibility with these surfaces. Finally, section 3.3 describes the implementation of MP(OR)U, our "Multi–Pipe Off–line Rendering Utility".

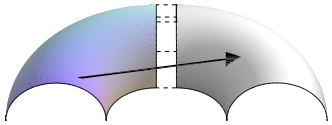
3.1 Parallel radiosity with complex surfaces

Basically, our new parallel radiosity algorithm proceeds as described in [7] with the improvements of [5]. We apply similar partitioning and scheduling techniques that allow to deliver an optimal load balancing — by minimizing idle time waiting on locks and synchronization barriers — while still exhibiting excellent data locality.

For simplicity reasons, we have chosen to keep a similar granularity by defining an elementary task as a whole radiosity transfer between two input surfaces. That is, a given process is in charge of computing a single radiosity transfer between an emitting input surface S^e and a receiving input surface S^r. At a given moment, two different processes might have to deal either with the same emitting surface S^e or with the same receiving surface S^r. In those cases, the *lazy copy* and *lure* (see fig. 2(a)) mechanisms introduced in [5] are exactly applied the same way. Actually, implementing complex surfaces into the previous algorithm could be done without rethinking the parallelizaton scheme, mostly thanks to the C++ object–oriented design of CANDELA, our radiosity research platform [28].



(a) When the receiving surface is locked, shoot is done on a lure.



(b) A self–shooting surface shoots on a lure.

Fig. 2. Using lures in parallel wavelet radiosity.

A minor difference is that the so defined elementary task is coarser compared to the previous parallel algorithm dealing with tessellated surfaces. This has led us to implement a sub–task stealing mechanism, that is activated either randomly (to avoid, mostly at the beginning of the computations, some tasks to take too long), or more specifically when the residual energy of the next emitting surface is much lower than the total residual energy currently being propagated in parallel. But this must be seen rather as an implementation than as an algorithmic contribution, at least until we undertake a deeper analysis.

A side–effect of the lure mechanism, firstly introduced for parallelism purpose, has helped dealing with the case where a curved input surface shoots energy onto itself. Handling self–shooting surfaces is a two–step process, as shown by fig. 2(b). The surface shoots energy on a lure as if it was a different surface. Then, the energy levels, residuals and mesh subdivisions of the lure are copied back to the original surface.

3.2 Hardware–based visibility with complex surfaces

As detailed in [2], our hardware–based visibility algorithm is an extension of the standard hemicube method [11] (see fig. 3) that removes its inherent reliability problems. It detects where the visibility cube is likely to provide the wrong answer and resorts to a classical ray–traced request in those places.

During the radiosity computations, this allows to quickly answer the two following kinds of visibility requests [8]:

- point to environment or surface to environment requests, *i.e.* what surfaces are potentially visible, from a point or a surface? These requests consist of “clipping” the subset of input surfaces visible from the current emitter;
- point to point requests, *i.e.* is a point visible from another point?

Obviously, complex shapes such as arbitrary planar surfaces or parametric surfaces

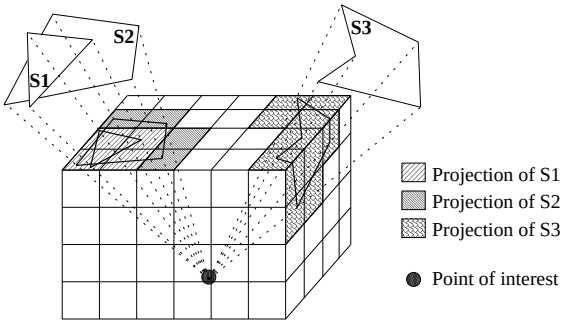


Fig. 3. The standard hemicube method.

can not be directly projected onto the graphics board. They have to be tessellated before being rendered. Note that this tessellation is totally unrelated to the resolution process. It only serves for the hardware–based visibility computations.

A unique "false" color is associated with *all* the triangles resulting from the tessellation of a given complex shape. Compared to the previous implementation where input surfaces were tessellated for the resolution process, this considerably reduces the risks of visibility errors. Indeed, a thin triangle resulting from a tessellation is no longer considered for visibility as a single entity but as an element of a set of triangles representing a complex shape: it is less likely to be missed due to the hemicube discretization.

If the hardware–based visibility algorithm can be directly applied to arbitrary planar surfaces (even including holes), care must be taken for curved surfaces. Indeed, a ray between a given point A and a point B located on a curved surface S (a sphere for example) may intersect another point C of the same surface S. In this case, the pixel intersected on the cube is the projection of the closest point between B and C, which is the only one visible from A. Then, when querying a pixel color on the hemicube, the associated Z–buffer value must be compared to the distance between the two considered points to check the validity of the answer.

3.3 MP(OR)U: Multi–Pipe Off–line Rendering Utility

In [8], we have designed a parallel radiosity algorithm that allows several graphics pipelines to perform the costly visibility requests while remaining computations are handled in parallel by multiple processors. At that time, the experiments we performed, although encouraging, were limited to a single graphics pipeline. We briefly present here the MP(OR)U programming interface we have developed to allow the implementation of the parallel algorithm over multiple graphics pipelines.

Description. This work is inspired by the SGI's MPU application programming interface, which allows several graphics pipelines to collaborate for the display, either of multiple sub–images of a same scene (for immersive environments), or of one single image (with a better frame rate or for larger scenes, depending on the chosen mode).

Basically, we have implemented the `MpxContext` and `MpxDrawable` classes over the OpenGL *context* and *drawable*. The key feature of these classes is that one instance of them is actually related to a set of graphics pipelines, preferably providing the *pbuffer* facilities for better performance [8]. Then when one want to use a `MpxDrawable`,

the object automatically tries to find an available graphics pipeline and, if it succeeds, makes the corresponding call to `glXMakeCurrent` for effective connection. The small overhead of the implementation is that we have to create a *display list* of the scene for each graphics pipeline: this slightly increases the initialization time at each modification of the input geometry.

The key functionalities of MP(OR)U include:

- A request to connect to a `MpxDrawable` may be blocking or not. When no graphics pipeline is available at the time of request, the process may either wait until one is finally acquired, or give up and return.
- A system of priority is implemented to sort the `MpxDrawable` connection requests. When a process makes a non–blocking request of a low priority, and that requests of higher priority are currently pending, it directly gives up and returns.

Example of use. We have taken into account the experimentations we have performed in [8] with one graphics pipeline to efficiently implement our parallel radiosity algorithm over MP(OR)U.

In particular, since visible surfaces clipping operations (see section 3.2) are critical in respect to the parallel execution, they have been assigned a high priority inside MP(OR)U, and the associated connection requests are obviously blocking. On the contrary, point to point visibility requests can be managed in software when no graphics pipeline is available, so the connection requests are non–blocking and have the lowest priority.

Finally, we try as far as possible to perform the visible surfaces clipping operations in advance, before the corresponding object becomes an emitter in the parallel solving computations. Otherwise, other processes having to deal with the same emitter would have to wait for the clipping operation to be completed.

4 Experiments

In all our experiments, we have used the same machine, a SGI Origin2000 with 64 MIPS 195 MHz R10000 processors and 24 Gbytes of main memory, connected to two SGI InfiniteReality2 graphics pipelines with 2 Raster Managers each.

4.1 The opera

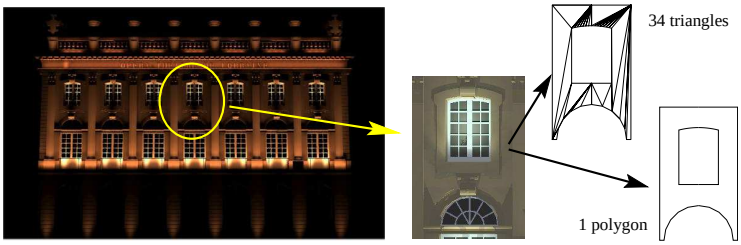
The *opera scene* is a faithful modeling of an opera house of the classical style designed by Emmanuel H´er´e (18^{th} century). Historically, this is one of the first architectural illumination project we have worked on, in collaboration with EDF (Electricit´e de France) and CRAI (Research Center in Architecture and Engineering). The original 3D model dates back to 1993, and has been created from architectural drawings with ARC+™ (from Design Labs[4]). At that time, the computations required the model to be divided in three parts (one for each floor), for separate computations requiring many hours each.

In this architectural model (as in many others), many polygons having non–trivial shapes appear. Up–to–now, these primitives had to be tessellated to be handled by radiosity algorithms. When using arbitrary polygons, each of these polygons is illuminated as a unique primitive. This is illustrated in fig. 4, which shows an image of a simulation of the *opera scene*.

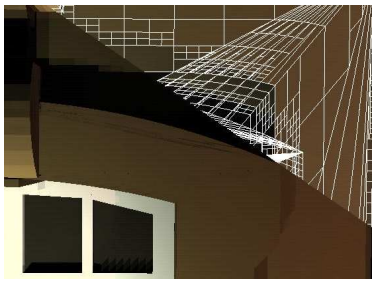
[4] `http://www.design-labs.com`

The original model has 32,429 triangles and parallelograms. When arbitrary polygons are allowed, the number of input primitives falls to 17,272. Allowing arbitrary polygons does not just divide the number of input primitives roughly by 2, it also significantly reduces the number of final patches (see Table 1). In addition, the number of surfaces that have shot energy during the simulation is lower by more than 25% for the same convergence rate (defined as the ratio of the energy already propagated to the initial energy). This is an important point which saves visibility computation time. Combined with the smaller number of input primitives, it explains why the overall computation time is roughly divided by 2.

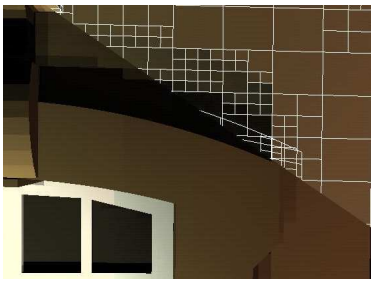
The differences between pre–meshing input primitives before radiosity computation and handling arbitrary polygonal primitives directly are illustrated in fig. 4(b) and fig. 4(c). False geometric discontinuities due to triangulation appear on fig. 4(b) along the shadows at the bottom left of the right window frame. Also, problems due to shapes not well suited to visibility computations can be seen above the window: long and thin triangles are missed by the visibility algorithm. By contrast, as shown in fig. 4(c) the extended domain algorithm produces a more regular meshing better suited to the approximation of the radiosity function.



(a) A close-up on the initial mesh of the opera: with or without arbitrary polygons.



(b) Radiosity solution without the use of arbitrary polygons



(c) Radiosity solution with the use of arbitrary polygons

Fig. 4. The opera house, modeled with ARC+^{TM}. By courtesy of EDF and CRAI.

scene	geometry	# input prim.	# final patches	comp. time	# shooting surfaces
Opera	ord. patches	32,429	819,109	476 s	9,358
	polygons	17,272	550,159	241 s	6,776
Soda Hall	ord. patches	272,450	1,124,585	44,585 s	9,010
	polygons	199,550	927,428	30,232 s	7,008
	quadrics	70,645	623,600	12,730 s	6,535
SH with geom. room	quadrics	71,957	827,627	13,411 s	6,642

Table 1. Numerical results for the illumination of the opera, the 4th floor of the Soda Hall and its geometry room, all for the same convergence rate of 95%, obtained with 48 processors and two graphics pipelines of the Origin2000.

4.2 The Soda Hall

The *Soda Hall scene* is a modeling of the famous computer science building on the campus of the University of California at Berkeley. The original dataset[5] is in the Berkeley proprietary UNIGRAFIX format[6], and we have converted it in the Open Inventor file format[7].

For our tests, we have computed the global illumination over the whole 4th floor with its furniture, using the $\mathcal{M}_3$ quadric wavelet basis . The original model of this floor contains 272,450 triangles and parallelograms, 216 point light sources and 905 lighting surfaces (including a neon modeled as a unique cylinder). When arbitrary polygons are allowed, the number of input primitives reaches 199,550. If in addition "curved" regions of the model (*i.e.*, regions where the mesh was originally meant to approximate curved objects) are modeled by quadric surface patches, then the number of input primitives falls to 70,645.

The numerical results found for the simulation of this floor of the *Soda Hall scene* confirm the results already observed with the *opera scene* (see again Table 1). The illumination of this large–scale model proves that it is much more efficient to handle complex shapes directly with our new techniques than through a tessellation. Between the ordinary patches version and the quadrics version of the 4th floor of the Soda Hall (number of input primitives is almost divided by 4), the number of final patches is roughly divided by 2 and the overall computation time is down by 70%.

4.3 The geometry room

The *geometry room* is a collection of geometric objects modeled with quadric surfaces, using SGDL[8], that we have included in a special room (420A for those in the know) of the 4th floor of the Soda Hall. It consists of a chess set, the well–known teapot, several benches and jars, and additional objects. Overall, this collection amounts to 1,312 surfaces, 549 of which are curved patches. The chess set alone – lit up by an additional desk lamp – is made of 555 surfaces, including 264 curved patches.

We have illuminated the entire 4th floor of the Soda Hall with this geometry room

[5] Available at `http://www.cs.berkeley.edu/~kofler`

[6] Described at `http://graphics.lcs.mit.edu/~seth/datasets/ug.fileformat`

[7] Available at `http://www.loria.fr/~cuny/SodaHall/sodaHall.html`

[8] `http://www.sgdl.com`

(see fig. 5). The inside view of the geometry room and the close-up on the chess set (see fig. 1) prove the realism of the simulation. Compared to the "traditional" Soda Hall model, the computation time of the simulation is only 5% larger. With previously known radiosity methods, it would have been impossible to simulate such a complex scene with a similar speed and accuracy. We have estimated that if the geometry room was uniformly tessellated with a grid size of roughly 2 inches, the number of input primitives would be multiplied by more than 60 and the computation time (for the sole geometry room) roughly by 4, for only a low visual accuracy.

Fig. 5. The Soda Hall. Original model in UNIGRAFIX. By courtesy of Carlo Sequin.

4.4 The virtual Reality Center™

As a final experiment, we present a full case study of a cooperation with SGI to build a virtual configuration software for Reality Centers. For this purpose, we have completely modeled the Reality Center located at SGI Cortaillod, using Maya™ modeling software (from Alias—Wavefront[9]).

The *Reality Center scene* is a perfect illustration of what a real model can be. Indeed, it is composed of a large set of quadric surfaces (the spherical screen, spherical walls, the conical desk, chairs made of ellipsoid and torus, *etc.*). The corresponding tessellated 3D models would contain more than 100,000 surfaces (for a medium approximation), while our model contains less than 3,000 objects.

As expected, our algorithm behaves very well with this scene. The execution time, with a single processor and no hardware–based visibility, is of about one hour. Figure 6 shows the speed–up curves for zero, one and two graphics pipelines, with one hour as the sequential reference time, even when hardware–based visibility is used.

First, we examine memory problems that could explain these curves. As in our previous work [7, 8] and whatever the machine configuration used, the data locality

[9] `http://www.aw.sgi.com`

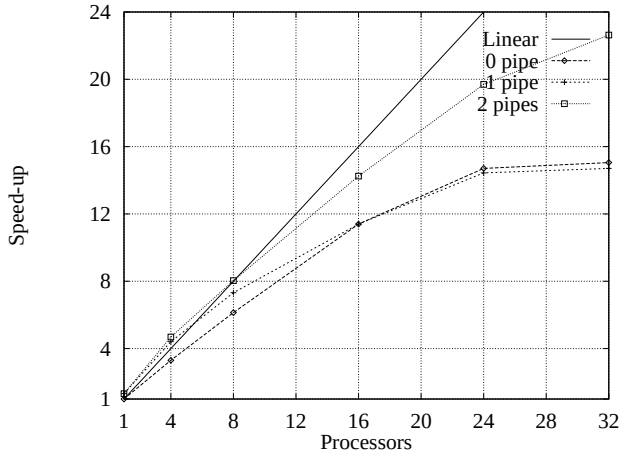


Fig. 6. Speed–up curves for the Reality Center™.

exhibited by the application is very high, since more than 95% of data accesses are satisfied in the L1 and L2 caches. Moreover, as explained in [6], using a larger virtual page size (1 Mbytes) than the default one (16 Kbytes) largely decreases the number of TLB misses and consequently the execution time. Finally, using the enhanced version of the LINUX/GNU libc allocator, also described in [6], guarantees contention free memory operations while keeping fragmentation low.

The speed–up curve when using the software–based visibility is not that good, since it remains under 16, even with 32 processors. The times lost on locks and on the final synchronization barrier do not explain this performance degradation. Actually, the `perfex` figures show an increase of the number of total cycles with the number of processors. This seems to be due to an overhead of our parallel algorithm that introduces supplementary work, when a process makes a lure on a surface that is actually totally occluded from the emitter. We will try, at the following tuning phase, to remove the cost of this unnecessary work.

Using one or two graphics pipelines gives a super–linear speed–up [10] with one and four processors. The benefit is not so tremendous compared to what it can be with the Soda Hall, mostly due to the smallest size of the model. Then, when the number of processors increases, it becomes less and less interesting to use a single graphics pipeline, compared to the software–based visibility. Indeed, we obtain the same time with 16 processors, and it is even worse to use a single graphics pipeline from 16 to 32 processors. This is due to the fact that the graphics pipeline becomes a huge bottleneck, and it is really critical for such short execution times (about 220 seconds).

Nevertheless, when using a second graphics pipeline, less time is lost to synchronize the use of graphics resources, and the obtained speed–up curve is much better (22 with 32 processors). This allows the whole *Reality Center scene* to be simulated in only 2 minutes. This definitely allows a semi–interactive rendering of the model. The user has then the opportunity to change the simulation parameters or the photometric and geometric characteristics of the input data, and to run the simulation again until he is totally satisfied. Figure 7 shows images of the obtained results.

[10] With the reference time being the execution time using the software–based visibility.

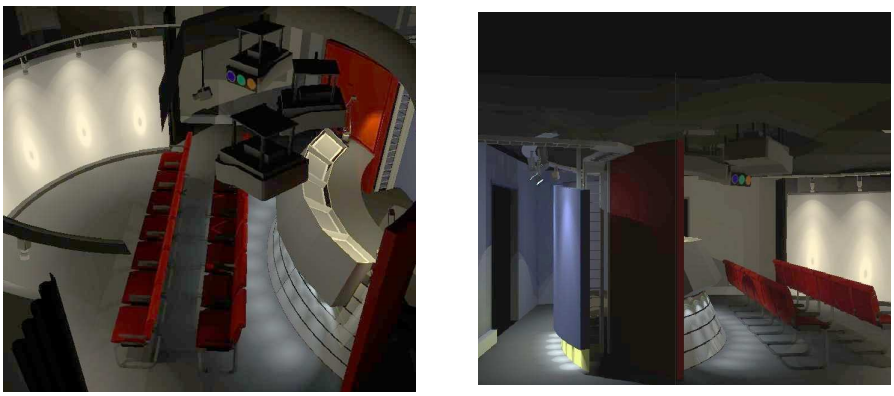


Fig. 7. The virtual Reality Center™, modeled with Maya™. Images by courtesy of SGI.

5 Conclusion and future works

In conclusion, we have presented a parallel and graphics hardware accelerated algorithm to conduct wavelet radiosity computations on scenes made of complex shapes. The algorithm relies on two innovative research works on high order wavelet radiosity — the extended domain algorithm and the constant Jacobian mapping — that remove the need to tessellate the complex input surfaces, resulting in more accurate computations, with faster convergence, and with smaller memory costs. The implementation intensively uses all available hardware resources, both processors and graphics pipelines, to reduce the computation times as much as possible.

In our future work, we want to continue to improve the scalability of our approach, in terms of both available hardware (processors and graphics pipelines) and size of input data (from the small single room to the whole building). We shall converge to an application that nearly allows real–time rendering of small radiosity solutions, and can also compute highly precise solutions of huge data sets in a reasonable time.

We also want to explore a combination of our algorithm with other efficient radiosity techniques, like clustering, to increase the size of the data our application can handle, or discontinuity meshing to increase the precision of the final solution.

Finally, we want to extend our algorithm to a greater number of parametric surfaces, such as Spline patches, B´ezier patches or NURBS, to be able to handle any kind of input surface. An orthogonal research direction may be to approximate these high–degree parametric surfaces by quadrics, as recently proposed in [19].

Acknowledgments

The authors would like to thank the Centre Charles Hermite for providing access to its graphics and parallel resources. We are also grateful to all the people that have contributed to the modeling of our test scenes and have given us permission to use them. Very special thanks to Jean–Claude Paul, that has started and motivated all this research on wavelet radiosity and high performance graphics, and to all the members of his fabulous ISA research team that have been involved in the success of CANDELA.

References

1. L. Alonso, F. Cuny, S. Petitjean, N. Holzschuch, and J.-C. Paul. Constant jacobian mappings: Accurate wavelet radiosity on parametric surfaces. Submitted to the *11th Eurographics Workshop on Rendering*, 2000. Available from http://www.loria.fr/~holzschu/Publications/jacobian.ps.gz.
2. L. Alonso and N. Holzschuch. Using graphics hardware to speed-up your visibility queries. *Journal of Graphics Tools*, 1999. Accepted for publication. Also available as http://www.loria.fr/~holzschu/Publications/99-R-030.pdf.
3. D. R. Baum, S. Mann, K. P. Smith, and J. M. Winget. Making Radiosity Usable: Automatic Preprocessing and Meshing Techniques for the Generation of Accurate Radiosity Solutions. *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, 25(4):51–60, July 1991.
4. K. Bouatouch and S. N. Pattanaik. Discontinuity Meshing and Hierarchical Multiwavelet Radiosity. In W. A. Davis and P. Prusinkiewicz, editors, *Proceedings of Graphics Interface '95*, pages 109–115, San Francisco, CA, May 1995. Morgan Kaufmann.
5. X. Cavin. Load Balancing Analysis of a Parallel Hierarchical Algorithm on the Origin2000. In *Fifth European SGI/Cray MPP Workshop*, Bologna, Italy, September 1999.
6. X. Cavin and L. Alonso. Parallel Management of Large Dynamic Shared Memory Space: a Hierarchical FEM Application. In *Seventh International Workshop On Solving Irregularly Structured Problems In Parallel (IRREGULAR'2000)*, Cancun, Mexico, May 2000.
7. X. Cavin, L. Alonso, and J.-C. Paul. Parallel wavelet radiosity. In *Proceedings of the Second Eurographics Workshop on Parallel Graphics and Visualisation*, pages 61–75, Rennes, France, Sept. 1998. Eurographics.
8. X. Cavin, L. Alonso, and J.-C. Paul. Overlapping Multi-Processing and Graphics Hardware Acceleration: Performance Evaluation. In *First Parallel Visualization and Graphics Symposium (PVG '99)*, San Francisco, California, October 1999.
9. A. Chalmers and E. Reinhard. *Parallel and Distributed Photo-Realistic Rendering*. ACM SIGGRAPH'98 Course Notes, July 1998. Course 3.
10. P. H. Christensen, E. J. Stollnitz, D. H. Salesin, and T. D. DeRose. Wavelet Radiance. In *Fifth Eurographics Workshop on Rendering*, pages 287–302, Darmstadt, Germany, June 1994.
11. M. Cohen and D. P. Greenberg. The Hemi-Cube: A Radiosity Solution for Complex Environments. In *Computer Graphics (ACM SIGGRAPH '85 Proceedings)*, volume 19, pages 31–40, Aug. 1985.
12. F. Cuny, L. Alonso, and N. Holzschuch. A novel approach makes higher order wavelets really efficient for radiosity. *Computer Graphics Forum (Eurographics 2000 Proceedings)*, 19(3), Sept. 2000. To appear. Available from http://www.loria.fr/~holzschu/Publications/paper20.pdf.
13. T. A. Funkhouser. Coarse-Grained Parallelism for Hierarchical Radiosity Using Group Iterative Methods. In *Computer Graphics Proceedings, Annual Conference Series, 1996 (ACM SIGGRAPH '96 Proceedings)*, pages 343–352, 1996.
14. S. Gibson and R. J. Hubbold. Efficient hierarchical refinement and clustering for radiosity in complex environments. *Computer Graphics Forum*, 15(5):297–310, Dec. 1996.
15. S. J. Gortler, P. Schroder, M. F. Cohen, and P. Hanrahan. Wavelet Radiosity. In *Computer Graphics Proceedings, Annual Conference Series, 1993 (ACM SIGGRAPH '93 Proceedings)*, pages 221–230, 1993.
16. P. Hanrahan, D. Salzman, and L. Aupperle. A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, 25(4):197–206, July 1991.
17. J. M. Hasenfratz, C. Damez, F. Sillion, and G. Drettakis. A practical analysis of clustering strategies for hierarchical radiosity. *Computer Graphics Forum (Eurographics '99 Proceedings)*, 18(3):C–221–C–232, Sept. 1999.
18. N. Holzschuch, F. Cuny, and L. Alonso. Wavelet radiosity on arbitrary planar surfaces. Submitted to the *11th Eurographics Workshop on Rendering*, 2000. Available from http://www.loria.fr/~holzschu/Publications/surfaces.ps.gz.

19. B. Lacolle and N. Szafran. Approximating parametric surfaces by means of quadric patches. In *Proc. of Fourth International Conference on Curves and Surfaces*, St. Malo, France, 1999.
20. D. Meneveaux, K. Bouatouch, and E. Maisel. Memory management schemes for radiosity computation in complex environments. In *Proc. Computer Graphics International '98 (CGI '98)*, pages 706–714, Los Alamitos, CA, June 1998. IEEE Computer Society Press.
21. S. Schaefer. Hierarchical radiosity on curved surfaces. In J. Dorsey and P. Slusallek, editors, *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 187–192, New York, NY, 1997. Springer Wien. ISBN 3-211-83001-4.
22. F. Sillion. A Unified Hierarchical Algorithm for Global Illumination with Scattering Volumes and Object Clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3), Sept. 1995.
23. J. P. Singh, C. Holt, T. Totsuka, A. Gupta, and J. Hennessy. Load Balancing and Data Locality in Adaptive Hierarchical N-body Methods: Barnes-Hut, Fast Multipole, and Radiosity. *Journal of Parallel and Distributed Computing*, 27(2):118, June 1995.
24. B. Smits, J. Arvo, and D. Greenberg. A Clustering Algorithm for Radiosity in Complex Environments. In *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)*, pages 435–442, 1994.
25. M. Stamminger, P. Slusallek, and H.-P. Seidel. Bounded radiosity - illumination on general surfaces and clusters. *Computer Graphics Forum (Eurographics '97 Proceedings)*, 16(3):C309–C317, 1997. Available from http://www9.informatik.uni-erlangen.de/eng/research/pub1997.
26. A. Willmott and P. Heckbert. An empirical comparison of progressive and wavelet radiosity. In J. Dorsey and P. Slusallek, editors, *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 175–186, New York, NY, 1997. Springer Wien. ISBN 3-211-83001-4.
27. A. Willmott, P. Heckbert, and M. Garland. Face cluster radiosity. In *Rendering Techniques '99*, pages 293–304, New York, NY, 1999. Springer Wien.
28. C. Winkler. *Expérimentation d'algorithmes de calcul de radiosité à base d'ondelettes*. PhD thesis, Institut National Polytechnique de Lorraine, 1998.
29. H. R. Zatz. Galerkin Radiosity: A Higher Order Solution Method for Global Illumination. In *Computer Graphics Proceedings, Annual Conference Series, 1993 (ACM SIGGRAPH '93 Proceedings)*, pages 213–220, 1993.