Wavelet Algorithms for Complex Models

François Cuny, Christophe Winkler, Laurent Alonso and Jean-Claude Paul
{Francois.Cuny,Christophe.Winkler,Laurent.Alonso,Jean-Claude.Paul}@loria.fr

Loria - Inria Lorraine,
Campus scientifique, BP 239,
54506 Vandœuvre-les-Nancy cedex, France

Abstract: We present the results of experimentations and tests with Wavelet Radiosity. We have developed a powerful wavelet radiosity implementation where we can independently modify every geometrical component of the scene (description of the input data, representation of spectral distribution, etc.) and every component of the global illumination algorithm (visibility algorithm, wavelet basis, etc.). This implementation has been tested on real world applications: an archaeological site reconstruction with daylight illumination, an opera house front with artificial illumination and the Soda Hall building inside illumination. In this paper, we present the results of our experiments, which are mostly about the interdependencies of the different parts of the general algorithm and the influence of each one on the final result. We also introduce several improvements to the wavelet radiosity algorithm that allow for higher rendering speed and lower memory use, thereby allowing rendering of architectural models of high complexity.

1 Introduction

Using hierarchical radiosity methods for solving the global illumination has proved to be the most efficient approach up to now. Nevertheless, its relatively low complexity in $O(k^2 + n)$, where k and n are respectively the numbers of initial input surfaces and final mesh elements, is still too high to make radiosity methods really practical in most real world situations. Clustering extensions, introduced by Smits *et al.* [9], address this weakness by creating higher level links between surface groups. This lowers the complexity to $O(k \log k + n)$, but introduces also some new problems due to the coarse visibility approximations.

Thus, we think that concurrently with research to improve these extensions, paying more attention to the implementation and parametrisation of the wavelet radiosity algorithms (which is the mathematical generalisation of the hierarchical approach), could be profitable.

Theoretical aspects of wavelet radiosity are discussed in [5, 8]. A few efficient implementations dealing with large environments have been proposed, based on simplification due to scene partitioning. Teller *et al.* [10] proposed an algorithm simplifying the visibility computations and handling scenes that do not fit in the main memory. Funkhouser [4] extended this algorithm to a coarse-grained parallel environment. He took advantage of the well-suited model of the Soda Hall. Its mainly axis parallel geometry and its photometric input data are indeed coarsely approximated. Such approximations are in fact rarely acceptable in many real world applications. Willmott and Heckbert [12] discuss various radiosity algorithms, their very interesting experimentation addresses many points of the general algorithms. But the tests related to wavelet radiosity algorithms are however restricted to a single option, the choice of the wavelet basis. Furthermore, except for the Haar basis, their wavelet-based algorithm breaks down when the input data consists of more than 1000 surfaces.

Analysing the behavior of the wavelet radiosity method is still an open problem, especially in real world applications. Two major problems appear. First, the numerous choices that affect in various ways the performance of the global algorithm, especially since they interfere with each other. The second problem comes from the software architecture which must be as flexible as possible. It must be possible to implement alternative algorithms and data structures. Furthermore, it must be robust enough to behave

correctly, even in limit cases. These options appear at different stages:

- the specification of the input data representing the geometrical and physical properties of the model;
- the discretisation techniques used for defining the functional space of the solution;
- finally, the computational techniques.

The global accuracy of the solution and the complexity of the resolution method depend on the errors introduced by all these choices.

We carried out experiments in order to have a better knowledge of the behavior of the wavelet radiosity method. This helped to design an efficient implementation and to understand the interdependence of the different algorithmic options. The performance criteria considered were the accuracy based on error control as well as on the empirical visual aspect of the final image, the computation time and the memory use. The test scenes include some classical radiosity-tailored scenes and scenes coming from real world applications. We also describe some improvements we made in order to be able to compute a solution for such scenes.

This paper is organized as follows. In section 2 we present a formulation of the radiance equation, and highlight the relation between the terms of the equation and the corresponding input data and computational methods. In section 3 we describe the software architecture we have designed in order to implement a set of algorithms. The experimentation protocol, including the performance criteria and test scenes, is defined in section 4. In section 5, we present the algorithms we implemented and the input data models we used. Finally, in section 6, we present our results and conclude.

2 Radiance Equation

Using some physical assumptions, the radiance equation models the propagation of light and its interactions with the surfaces of the environment. Rather than resorting to the usual mathematical presentation of the radiance equation, we shall here relate the different quantities to the physical properties and algorithms used to take them into account. The equation can be written as:

$$f(\lambda, x, \omega^{\uparrow}) = g(\lambda, x, \omega^{\uparrow}) + \int_{\Omega} f(\lambda, y, \omega^{\downarrow})\, k(\lambda, x, \omega^{\downarrow} \rightarrow \omega^{\uparrow})\, d\,\mu,\ \ where: \tag{1}$$

- $\boldsymbol{f}$, the *radiance function*, is the solution that must be computed as precisely as possible through the simulation process. This function is defined on any point x of the input *surface set* $\mathcal{M}$. The type and shape of these surfaces affect the resolution method. They define the geometric support of the basis functions used to approximate f.
- $\boldsymbol{g}$, the *emission function*, represents the initial radiance, usually associated with the light sources. The expression of this function strongly influences the final result. In fact, the most important energy propagation corresponds to the direct illumination.
- $\boldsymbol{k}$ is known as the *bidirectional reflection distribution function (brdf)*. It characterises the physical properties of the materials in the environment and defines the behavior of the reflection of light.
- $\boldsymbol{\lambda}$ is the *wavelength* at which given functions are computed. The functions f, g and k are all defined as *spectra*, *i.e.* continuous functions of the wavelength.
- $\boldsymbol{y}$ is the nearest point of the environment visible from x in the direction $-\omega^{\downarrow}$. It can be defined as: $y = x - v(x, \omega^{\downarrow})\omega^{\downarrow}$ where the implicit *visible-surface function* $\boldsymbol{v}$ returns the distance of the nearest point from x in direction $\omega^{\downarrow}$. If such a point does not exist $v(x, \omega) = \infty$. The visible-surface function has a central role in the resolution process, because it causes discontinuities in the radiance function and its derivatives. Furthermore, it is also the most expensive algorithmic part in computation time.

Choices affecting these parameters must be considered with care, since they will affect the final result.

Other choices must still be done to solve equation 1. The radiance function must be projected onto a finite space of functions: $f \simeq \tilde{f} = \sum_{i=0}^{n} \alpha_i \phi_i$ where the ϕ_i functions must define a wavelet basis in order to perform wavelet radiosity.

One way to perform the resolution using a finite element method is to use a Galerkin method. This yields a linear system that can be solved:

$$\begin{bmatrix} a(\phi_1,\phi_1) & a(\phi_1,\phi_2) & \cdots & a(\phi_1,\phi_n) \\ a(\phi_2,\phi_1) & a(\phi_2,\phi_2) & \cdots & a(\phi_2,\phi_n) \\ \vdots & \vdots & \ddots & \vdots \\ a(\phi_n,\phi_1) & a(\phi_n,\phi_2) & \cdots & a(\phi_n,\phi_n) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} < g,\phi_1 > \\ < g,\phi_2 > \\ \vdots \\ < g,\phi_n > \end{bmatrix}, \quad (2)$$

where the $a(\phi_i,\phi_j)$ are the kernel coefficients associated with the integral and $< \cdot,\cdot >$ is the scalar product in the functional Lebesgue space $\mathcal{L}^2$. This system can be so huge that it is impossible to look for an exact solution.

At this stage, we still need to:

- choose the *basis functions* ϕ_i, to form an efficient basis to represent f,
- evaluate the *coefficients* $a(\phi_i, \phi_j)$,
- choose a method to solve the *linear system.*

Arvo *et al.* [1] analysed equation (1) and quantified the errors introduced by all these approximations. This work is very usefull when you start programming radiosity applications since it provides many theoretical knowledge the algorithmics choices. But, as it was carried out at an abstract level it can hardly infer usable information for real-world applications.

For our applications we use the radiosity assumption which supposes that every surface is perfectly diffuse, according to the Lambertian model. This simplifies equation (1) because in this case the functions defined above are independent of the incoming ($\omega^{\downarrow}$) and outgoing ($\omega^{\uparrow}$) directions.

3 Software Architecture

We have designed our software architecture so as to be as flexible, evolutive and robust as possible [6]. The use of C++ allows us to easily modify or extend the implemented classes. Our goal was to design a simulation environment well suited to the description of the geometric and radiometric properties of the scene, the mathematical representation of the problem to resolve, and its expression in terms of algorithms and data structure. Note that the analysis done in [1] was very helpful for the conception of the architecture (figure 1).

- **Scene Graph.** We have chosen to represent the scene by a hierarchical description, based on an OpenInventor graph [11]. We kept the graph traversal mechanism of OpenInventor. Roughly described, an *action* performs a depth-first traversal of the tree, and the encountered property nodes only affect the subsequently traversed nodes. Thus, different nodes can share the same properties. This allows us to define our own nodes representing the input data associated to $\mathcal{M}$, g, k and λ, but also to the basis functions ϕ_i.
- **Algorithmic Graph.** In order for our system to be as open as possible, we designed algorithmic classes. The global resolution algorithm is built together from a combination of algorithmic plug-ins. As for the scene description, the algorithm is defined by a tree called *script*. This allows a dynamic instantiation of the plug-ins, and the applied resolution method can be modified by changing the file containing the *script* independently of the software compilation. These algorithmic nodes are activated when traversed by a specific *action*. For example, a node will build a space partition to accelerate visibility computations, another will perform resolution steps based on its own parameters and on algorithmic parameters found in the scene graph.
- **Algorithmic Parameter Nodes.** Some algorithm-dependent parameters and even algorithmic plug-ins like kernel computation methods may vary within the scene. For example, it can be useful to simulate the part of the scene that we want to visualize with a good precision. Meanwhile, in order to account for the influence of the rest of the scene, the solution must also be computed there, but it can be less accurate. This can be done by setting different kernel error bounds or kernel computation methods in the scene graph. This behavior is somewhat like the overloading mechanism in C++.

The flexibility provided by the algorithmic tree description is original and is a major asset of our software architecture.

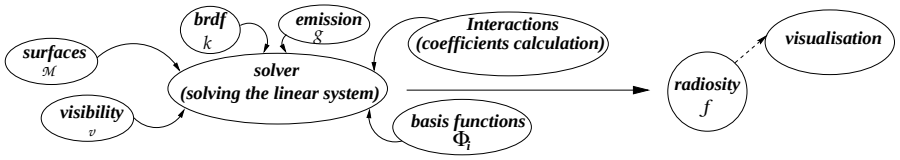


Fig. 1. General overview

4 Experimentation methods

We handled all the experiments with our own platform on one and the same machine (a Power Onyx RE2 with a processor R10000 (194 MHz) and 512 Mo of RAM). Thus, hardware differences and programming style should not influence our results. We adopted an experimentation protocol similar to Willmott *et al.* [12], with only slight differences. The wavelet radiosity algorithms we want to compare are iteratively converging over time. We measured information relevant to the computations at different **checkpoints**, allowing statistics and algorithmic comparison over the time. This will help comparing the evolution of the algorithms.

4.1 Performance criteria

• **Numerical error control.** We chose to use the residual to control our error. This view-independent criterion gives a precise idea of how far the simulation process has converged. One main advantage of this approach is that it does not rely on any reference solution, *i.e.* it does not need any precomputation. We do not use the root mean square error measure. It needs an exact reference solution that does not exist except for a few extremely simple scenes. The reference solution is usually computed with a radiosity of a Monte Carlo method, thus introducing errors. Furthermore, this causes some real feasibility and precision problems for large and complicated scenes.

• **Visual error control.** It is very important for most real world applications to compute acceptable looking final images. Furthermore, theorical error controls can not account for all the visible artefacts. Thus we use architects users experience for setting numerical error control and to have a visual accuracy.

• **Memory use.** Since we want to check algorithms general efficiency we measure the actual memory used for each computation. We set the numerical error control so that the memory needed for each algorithm could not prevent from completing computations.

• **Computation time.** We make comparition of complete computation time (stopped for the same numerical error control) for each algorithms with the same parameters.

4.2 Experiment scenes

While we focused on large scenes coming from real world applications, we also used one radiosity-tailored simpler scene in order to highlight some results. Some scenes are really suited to isolate specific behaviours of the radiosity algorithms. Depending on their specific characteristics, we were able to fix individual parameters and to estimate their influence on the algorithmic behavior.

• **Class room** this is the reference test scene number 8 made available by Peter Shirley for the 5^{th} Eurographics Workshop on Rendering. In this room, the chairs are not all identical in size, nor axis aligned. This model was designed for radiosity algorithms: all its 3138 surfaces are rectangles.

This is rarely the case for models coming from real world applications where the input surfaces are often concave, or even have holes. Usually, the data base comes from geometric modelers, or also from digitisers taking laser measures. Thus, the geometrical complexity of a scene depends on the number of surfaces, and also on the complexity of each surface.

Moreover, such scenes contain often extremely accurate values of photometric data for light sources and of spectral properties of surface materials. Usually, these data are physical measures taken by gonio-photometers and spectro-photometers. The spectra of real artificial light sources are often discontinuous,

with high emision peaks. Usual radiosity methods performing the computations for a few wavelengths will typically miss these peaks. It is crucial to account precisely for these input data to compute accurate solutions.

• **Stanislas Square Opera** in Nancy. The main goal of this experiment was to simulate new lighting designs. The model was created from architectural drawing analysis, and it was thus not specifically prepared for illumination. It represents the front of an opera house thus with very few interreflections. The precision of the solution highly depends on the light and reflectance models used for the resolution.

• **Delphes site temple** of the Marmaria sanctuary. We had to reconstruct the antique greek archeological site. The archeologists built a highly detailled geometric model from very various data sources, including digitisers which provided many triangles. The images had to be computed with natural light sources. For simplicity, we did not implement specific models for the sun and the sky lights. Thus, we made the assumption of a parallel light source because we wanted to focus on interreflections.

• **Soda Hall**. It was selected because it is a reference scene. Except for the room furnitures, the geometric model is highly axis-aligned. This scene is suitable for virtual reality applications and interactive walk-through. We chose this scene to demonstrate that it can be illuminated one floor with furniture, even without relying on partitioning and grouping algorithms as in [10, 12].

5 Implemented Algorithms

5.1 Geometry representation, $\mathcal{M}$

• **Complex input polygons.** The surfaces define the support of the radiosity function. Classical wavelet approaches handle only triangles, rectangles or parallelograms. This allows to use precomputed push-pull coefficients, but implies triangulating the geometric data base, causing a multiplication of the surfaces. One possibility to deal with general polygons is to map the radiosity function on the polygon, and to design a subdivision method. This implies to compute and to store all the push-pull coefficients between any two levels of wavelets. The storage cost can be a major drawback if the links are not stored.

We designed an alternate original approach accounting for any kind of polygon, even with holes. We use the same algorithm as for parallelograms (same subdivision and push-pull values), except that we extend the radiosity function on the bounding parallelogram of the surface. And when the surface emits, we only use the radiosity that actually resides on the original polygon.

• **Non-standard decomposition.** For wavelets defined on parallelograms, the usual subdivision between two level is a quad-tree cut. The ratio between the width and the length of the polygons remains the same at every level.

We chose to cut a surface into four widthwise pieces if the ratio is larger than 2. This helps dealing with thin parallelograms by reducing the variation of the function on the mesh. Hence we need less subdivision levels.

5.2 Light source models, g

Light source models must be considered with great attention, because they are responsible for the major energy transfers. In order to treat light sources with greater precision, we choosed to separate the direct illumination from the interreflection phase.

• **Artificial light sources.** Diffuse sources can be described with emitter patches. In reality, they emit energy according to a spatial distribution. We use C/γ and B/β data models to be as accurate as possible. Such values can be retrieved from the light source manufacturer. Physical measurements done with gonio-photometers can also be expressed with these models without loss of information.

• **Natural light sources.** Actually, we approximate the sun light with a parallel light source at infinite distance, and do not account for the sky light.

5.3 Reflectance function, k

The radiosity assumption supposes that every surface is diffuse, so we only use the ideal diffuse model for the reflectance function.

5.4 Spectral distribution, λ

The spectral emittance of light sources (g) and the reflectance properties materials (k) are defined by spectral distributions. Accurate models for these distributions must be considered to avoid missing among others the influence of narrow emission peaks (up to $5nm$) of artificial lights. We implemented a large set of spectral basis functions and defined the interactions between spectra, that is addition, multiplication and projection onto another basis. Hence, we can choose to process the radiosity computations in any spectral basis.

5.5 Visibility computation, v

The visibility computations are extremely time consuming. It has thus been natural to look for accelerations. Three different methods are available:

- **Z-buffer visibility**, using the graphical hardware. Visibility requests are answered by projecting the scene on an off-screen bitmap which is then analysed.
- **Ray-tracing visibility**, accelerated by using a hierarchical space partition. The subdivision implemented [3] is very appropriate for architectural scenes.
- **Hybrid visibility**, a compromise between the above two algorithms. Instead of relying on one single pixel, we also consider its neighbours. If they relate to two different surfaces, we suspect visibility problem. In this case we choose either to zoom in, or to use ray-tracing.

5.6 Wavelet function basis, ϕ_i

The different implemented basis are the $\mathcal{M}_1$ (constant Haar basis), $\mathcal{M}_2$ (three linear functions for triangles, with one more bilinear function for parallelograms) and $\mathcal{M}_3$ wavelet basis (six quadric functions on triangles and three more cubic functions for parallelograms).

5.7 Kernel coefficient calculation, $a(\phi_i, \phi_j)$

We chose to use the Galerkin method to define the approximated radiosity function. The reason is that this approach minimizes the mean error of the function over its support. We think that it is more appropriate than the point collocation method which minimizes the error at a given set of points. The Galerkin method defines the expression of the kernel coefficients which are computed by using a Gauss quadrature method. This allows us to remain consistent with the wavelet functions.

5.8 Solver method

In order to solve the linear system of equations 2, we have implemented the classical wavelet gathering method (*Gauss-Seidel*). This method stores the links between the interacting surfaces. Unfortunately, the memory needed to store the links makes this approach unpractical for large scenes.
In order to overcome this problem, we adapted the *Southwell* algorithm with the physical intuition introduced by Cohen [2] to have a progressive wavelet algorithm. In order to overcome the memory cost due to the link storage, we can choose to store the links and refine them when needed or to discard them and recompute the kernel on the fly every time a surface reemits. A third option is to store only the top level of the links, in order to retrieve easily the surfaces which are visible from an emitter. This is an inbetween solution, keeping the memory use acceptable, without needing too much computation time.

5.9 Complete computation process

All the choices described above are independent in our software architecture hence it is possible to test any combination of those methods. This approach made it possible for us to test an important variety of complete resolution processes. The main difficulty of designing efficient global illumination processes lies in finding a good compromise between the many algorithmic options so as to maximize performance. Indeed, the effects of one choice depend on all other choices since the algorithm does not rely sequentialy on these choices.

6 Results

We designed tests in order to highlight the influence of every algorithmic option. This helped to have a better understanding of the behavior of the radiosity methods.

6.1 Inputs

• **Geometry of the surfaces:** The common method used to represent complex surfaces in the scene is to split them into triangles and rectangles. We have the possibility to directly compute radiosity values on complex polygons such as concave polygons and polygons with holes. This allows us to have a smaller number of polygons in our computations, which also better fits the wavelet basis. Moreover it saves us a triangulation step which would result in numerous triangles, possibly pathological. Such complex polygons frequently occur in scenes coming from geometrical modellers or laser measurements.
The modellers do also often generate very thin polygons. To deal with such thin polygons, we introduced a non-standard splitting that ensures an improvement of the aspect ratio. With this non-standard splitting, we can obtain results of similar quality with a shallower refinement.
In figure 4 you can see meshing and illumination done with such complex polygons. As one can see in figure 2, using such geometrical advanced algorithms reduces by 33% to 66% the number of meshes created and by 66% to 72% the computation time.

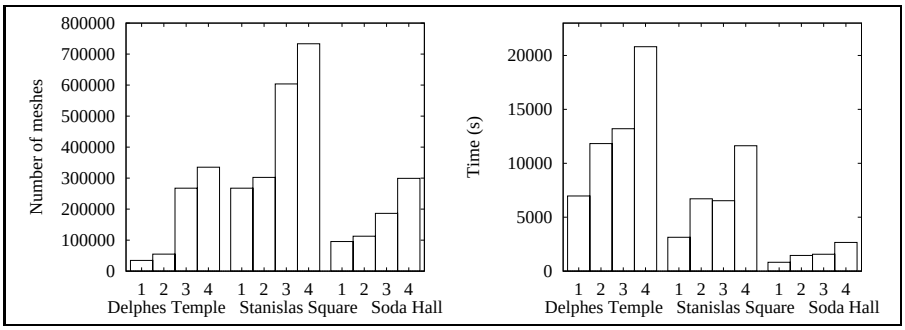


Fig. 2. Effect of the use of complex polygons, 1: not triangulated/ non-standard, 2: triangulated/ non-standard, 3: not triangulated/ standard 4: triangulated/standard

• **Light sources:** Direct illumination is of great signification in global illumination simulation especially for artificially illuminated scenes. In scenes such as the Stanislas Square Opera model and the Soda Hall model direct illumination corresponds to the greatest exchange of energy and also to the most visible effects in the simulation like shadows and highlights. Using emitter patches as light sources for these scenes gives very poor results. Moreover technical spatial attributes of light sources are usually available in C/γ measurements. These are the reasons why we are using them directly in our computations.
Besides, we are generally dealing with a great number of light sources (about a thousand for the Soda Hall model and nearly one hundred for the Stanislas Square Opera model, and many more if we consider the sky light as a set of lighting elements). For both these reasons we have separated direct illumination from the subsequent iterations. Figure 5 shows the quality and the realism achieved by our simulation.

• **Spectral distribution:** In order to take better into account the spectral properties of the scene, one needs to perform the computations with more spectral data.
Tests prove that using a spectral basis with 81 basis functions coeffi cients[1] requires about 1.5 times more computation time than Meyer's classical approach [7], but of course it also uses more memory space (14 times more than Meyer with four basis functions coeffi cients, because of space storage optimisation).
Thus, mainly for memory use considerations, we must try to avoid the use of detailed spectral values during computations. This can be done by restricting their use on parts of the scene where the spectral properties are not smooth.

6.2 Algorithms

• **Visibility computation:** We have implemented three different visibility algorithms:

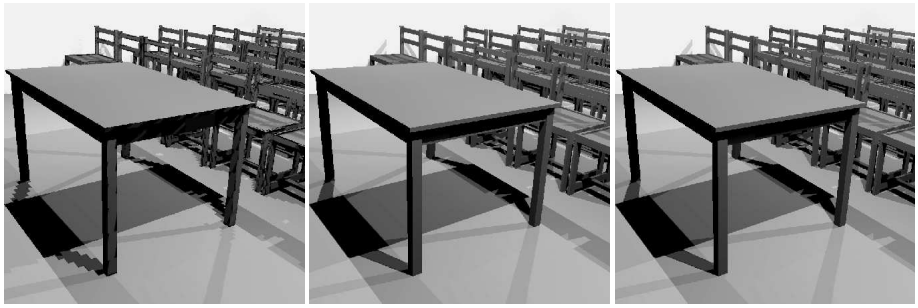
- Z-Buffer visibility: this method of visibility is very quick but gives poor results (jagged shadow boundaries). They are inherent to the sampling process, and depend on the pixmap size. The major drawback is that the visibility errors are very large near the junction of two surfaces which leads to greater subdivisions near the junction.

  An unexpected result is the higher number of fi nal mesh elements than for the other approaches. The reason is that these elements must fi t the highly jagged edge of the shadow boundaries.

- Ray-tracing visibility: this visibility gives exact point-to-point results, but of course it is slower than the fi rst one.

- Hybrid visibility: most of the time, this method gives the same result as the ray-tracing visibility method and its computation time is always smaller. Of course it spends more time than the Z-Buffer visibility but the rate between the extra-time and the improvement of the quality appears to be very good.

Numerical results are given in table 1 and the associated images are in fi gure 3.



(a) Z-Buffer (b) Ray-Tracing (c) Hybrid

Fig. 3. Effect of the visibility algorithm

The Z-buffer visibility algorithm is well suited to fi nd the runtime parameters, whereas the hybrid method is clearly the more effi cient approach to compute very accurate solutions.

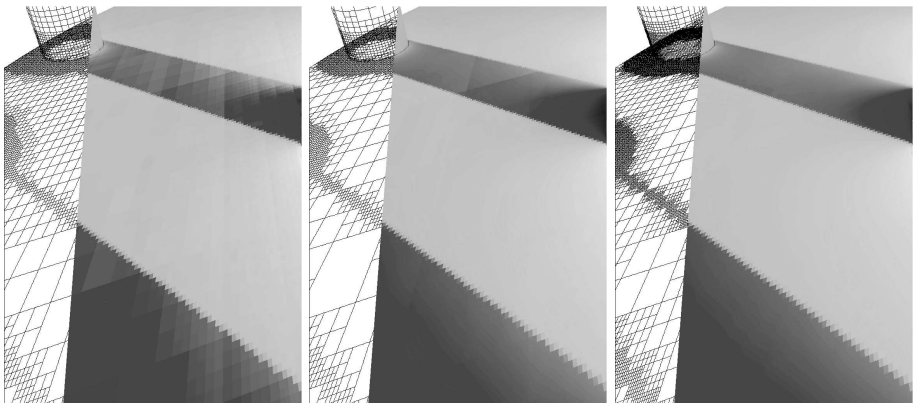
• **Wavelet basis functions:** We have tested the fi rst three multi-wavelet basis: the $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ wavelet bases.
In the case of constant lighting (like for Tholos site temple illumination), Haar basis can very well approximate the radiosity functions.

[1] This corresponds to the most accurate spectral data we got from physical measures.

Visibility algorithm	Z-Buffer	Ray-Tracing	Hybrid
Number of final meshes	1 089 982	740 734	711 034
Computation time (s)	1 222	4 764	1 978

Table 1. Shirley class room

However a noticeable thing is that the $\mathcal{M}_2$ basis and the $\mathcal{M}_3$ basis give more satisfying results when some occlusion problems appear during shooting between two polygons (figure 4). The approximation quality criterion of our oracle relies on the $\mathcal{L}^\infty$ norm of the difference between the received energy before and after projection in the wavelet basis. In the case of partial occlusion, the higher-order bases will exhibit a larger divergence at certain places. This induces a finer subdivision in these areas, which strongly enhances the visual quality of the result. Getting the same quality with Haar bases would need to take into account a much smaller refinement criterion which would also lead to a larger computation time.



(a) Haar wavelet (b) Linear wavelet (c) Quadric wavelet

Fig. 4. Effect of the wavelet basis

With artificial light sources, higher-order bases naturally catch the variations of the radiosity function on illuminated polygons. Most of the time the linear basis sufficiently matches the variations of industrial light sources and the quadric basis is only fully used when really precise computations are needed. Table 2 shows the effect of the wavelet basis choice for the three reference architectural models.
Even if Haar bases are often less memory and time consuming, the $\mathcal{M}_2$ or $\mathcal{M}_3$ bases seem more appropriated to give good visual quality results.

• **Solver method:** We started with implementing the standard gathering method that gave good results on moderately large scenes (less than 500 input polygons). On a larger scene with many subsequent light bounces, we found it impossible to complete the computation. Therefore we implemented a progressive wavelet shooting method that did not have to store the links. This method proved to be very efficient on fairly large scenes.
We tested it on:

- scenes with many light sources, for the Soda Hall and the Stanislas Square Opera (about one hundred light sources),
- scenes with important subsequent iterations, for the Tholos site temple (6 237 shooting steps

Scene	Wavelet used	Number of final meshes	computation time (s)
Tholos site temple	Haar	37 114	1 668
	$\mathcal{M}_2$	30 246	2 548
	$\mathcal{M}_3$	29 434	12 802
Stanislas Square Opera	Haar	404 787	683
	$\mathcal{M}_2$	162 123	594
	$\mathcal{M}_3$	152 863	763
Soda Hall	Haar	1 492 624	3 401
	$\mathcal{M}_2$	621 312	3 066
	$\mathcal{M}_3$	618 192	4 135

Table 2. Wavelet basis influence

which represents more than 40 525 848 interactions between surfaces after direct illumination),

– scenes with a huge geometrical data base, especially for the Soda Hall (954 868 polygons after computation)

6.3 Complete computation process

All the experiments we described before lead us to define a general efficient complete algorithm for the radiosity computation on three architectural models. The best configuration seems to be: handling complex polygons with non-standard decomposition; representing the spatial distribution of the light sources with C/γ models; solving the system with the progressive shooting approach without storing the links; using the $\mathcal{M}_2$ wavelet basis; and computing visibility with our hybrid algorithm.

With this configuration, we achieve the simulation of the three real world scene:

• Tholos site temple (7 778 input polygons, 604 742 final meshes in a 22 h CPU time for a complete time of 23 h, with a maximum memory use of 244 Mo),

• Stanislas Square scene (34 614 input polygons, 78 light sources, 405 568 final meshes in a 1 h 45 min CPU time for a complete time of 1 h 46 min, with a maximum memory use of 198 Mo. These figures only account for the direct illumination),

• Soda Hall scene (143 097 input polygons, 163 light sources, 954 868 final meshes in a 12 h 23 min CPU time for a complete time of 38 h 49 min, with a maximum memory use of 822 Mo. These figures only account for the direct illumination).

You can see pictures of the results in colour plate (figure 5).

7 Conclusion

In conclusion, we have determined the following results:

- Complex geometrical and physical data representation is essential for lighting computations in real world scenes.
- A shooting algorithm without link storage can be used to compute the global illumination simulation on large scenes using wavelets. Surprisingly, Willmot *et al.* [12] did only consider wavelet radiosity with gathering and storing links. Our experimentations show that the progressive shooting method can indeed be used with wavelet radiosity, and that it saves memory space and ensures a good convergence of the algorithm.
- The $\mathcal{M}_2$ wavelet basis is generally the best to perform global illumination with a good quality. Except in the case where there is constant lighting on the surfaces, the computation with $\mathcal{M}_2$ wavelet basis required less memory space and less computation time than with the Haar and the $\mathcal{M}_3$ bases.

- An advanced visibility algorithm using hardware and space partitioning is essential to illuminate large scenes with occlusions. Our experimentation showed that visibility computations in complex environment represents in themselves about 50% of the whole computation time. Furthermore, a crude algorithm can in fact *increase* the number of decompositions where there are shadow boundaries while the hybrid algorithm combines both speed and accuracy.

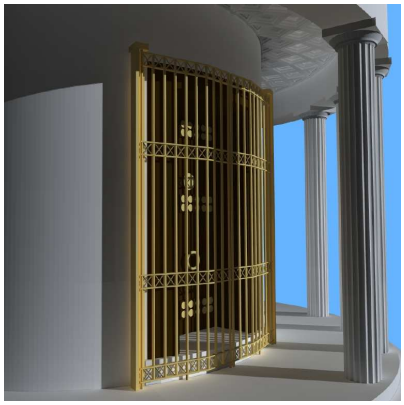
Combining all these methods contributes to achieve global illumination simulation on quite large scenes of various origins.

Acknowledgments

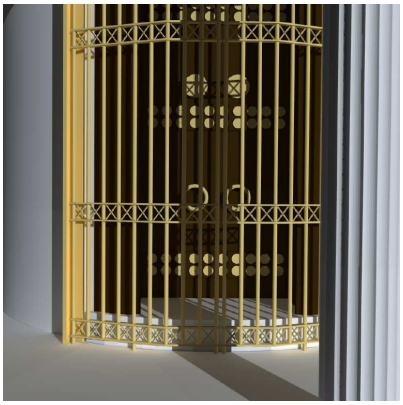
The authors would like to thank Nicolas Holzschu, Slimane Merzouk and Xavier Cavin from the ISA team of the LORIA (`http://www.loria.fr`) for their work on Candela, Didier Bur and Bertrand Courtois from the school of Architecture of Nancy (`http://www.crai.archi.fr`) for their work on the models, Marc Albouy, from Electricité de France which provided the Stanislas Square, Carlo Sequin who provided the Soda Hall.

References

1. James Arvo, Kenneth Torrance, and Brian Smits. A Framework for the Analysis of Error in Global Illumination Algorithms. In *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)*, pages 75–84, 1994.
2. Michael Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A Progressive Refinement Approach to Fast Radiosity Image Generation. In *Computer Graphics (ACM SIGGRAPH '88 Proceedings)*, volume 22, pages 75–84, August 1988.
3. M. de Berg. Linear size binary space partitions for fat objects. In *Proceedings of the 3rd Annual European Symposium on Algorithms*, pages 252–263, 1995.
4. Thomas 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.
5. Steven J. Gortler, Peter Schroder, Michael F. Cohen, and Pat Hanrahan. Wavelet Radiosity. In *Computer Graphics Proceedings, Annual Conference Series, 1993 (ACM SIGGRAPH '93 Proceedings)*, pages 221–230, 1993.
6. Slimane Merzouk. *Architecture logicielle et algorithmes pour la résolution de l'équation de radiance*. Ph.D. thesis, Institut National Polytechnique de Lorraine, Vandœuvre-lès-Nancy, France, July 1997.
7. Gary W. Meyer. Wavelength selection for synthetic image generation. *Computer Vision, Graphics and Image Processing*, 41:57–79, 1988.
8. Peter Schroder, Steven J. Gortler, Michael F. Cohen, and Pat Hanrahan. Wavelet Projections for Radiosity. *Computer Graphics Forum*, 13(2):141–151, June 1994.
9. Brian Smits, James Arvo, and Donald 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.
10. Seth Teller, Celeste Fowler, Thomas Funkhouser, and Pat Hanrahan. Partitioning and Ordering Large Radiosity Computations. In *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)*, pages 443–450, 1994.
11. Josie Wernecke. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2*. Addison-Wesley, Reading, Massachusetts, 1994.
12. Andrew J. Willmott and Paul S. Heckbert. An empirical comparison of radiosity algorithms. Technical Report CMU-CS-97-115, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, April 1997. Available from http://www.cs.cmu.edu/ radiosity/emprad-tr.html.



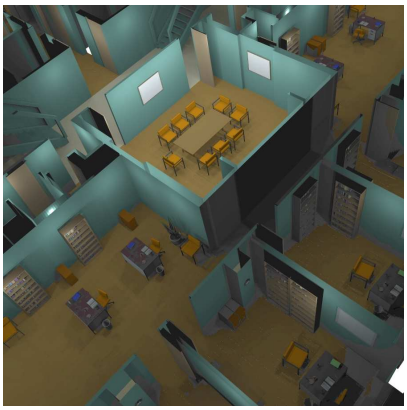
(a) Tholos temple simulation



(b) Another view of Tholos temple

(c) Stanislas Square photograph

(d) Stanislas Square embedded simulation



(e) Soda Hall simulation

(f) One room of the Soda Hall

Fig. 5. Images generated with our algorithms