The Virtual Mesh: A Geometric Abstraction for Efficiently Computing Radiosity

Laurent Alonso, François Cuny, Sylvain Petitjean, Jean-Claude Paul, Sylvain Lazard, Eric Wies

The Virtual Mesh: A Geometric Abstraction for Efficiently Computing Radiosity

L. Alonso, F. Cuny, S. Petitjean, J.-C. Paul, S. Lazard and E. Wies
Loria - Inria Lorraine

In this paper, we introduce a general-purpose method for computing radiosity on scenes made of parametric surfaces with arbitrary trimming curves. By contrast with past approaches that require a tessellation of the input surfaces (be it made up of triangles or patches with simple trimming curves) or some form of geometric approximation, our method takes fully advantage of the rich and compact mathematical representation of objects. At its core lies the *virtual mesh*, an abstraction of the input geometry that allows complex shapes to be illuminated as if they were simple primitives. The virtual mesh is a collection of normalized square domains to which the input surfaces are mapped while preserving their energy properties. Radiosity values are then computed on these supports before being lifted back to the original surfaces.

To demonstrate the power of our method, we describe a high-order wavelet radiosity implementation that uses the virtual mesh. Examples of objects and environments, designed for interactive applications or virtual reality, are presented. They prove that, by exactly integrating curved surfaces in the resolution process, the virtual mesh allows complex scenes to be rendered more quickly, more accurately and much more naturally than with previously known methods.

Categories and Subject Descriptors: I.3 [**Computing Methodologies**]: Computer Graphics; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Radiosity, wavelets, illumination, curves & surfaces, virtual mesh

1. INTRODUCTION

Since radiosity techniques have interesting properties that make them useful for new applications (simulation of light to give a filmic "look" to computer animations, precomputed environments for virtual reality), a direct rendering of scenes made of complex shapes becomes highly desirable. Scenes created with modeling software typically include parametric surfaces such as NURBS, cylinders, spheres, toroidal patches, arbitrary planar primitives and their CSG combinations. Moreover, the limit surfaces of many popular subdivision schemes (such as Loop or Catmull-Clark) are known to be parametric surfaces – see, e.g., [Stam 1998]. This paper presents a novel and powerful method for the direct radiosity-based rendering of such surfaces.

One assumption of the original radiosity algorithm is that every object of a scene is made of a collection of planar patches [Goral et al. 1984]. Unfortunately, few of the

This paper builds upon a Eurographics rendering workshop paper [Holzschuch et al. 2000].
Address: Loria, BP 239, 54506 Vandœuvre cedex, France
{alonso,cuny,petitjea,paul,lazard,wies}@loria.fr
http://www.loria.fr/isa

Fig. 1. This geometry room has been illuminated with a high-order wavelet radiosity solution using the virtual mesh. The curved objects have been modeled with the SGDL modeling kernel (`http://www.sgdl.com`).

later improvements have entirely freed themselves of this hypothesis. For instance, in hierarchical radiosity [Hanrahan et al. 1991], lower-level elements are assumed to be planar polygons. Galerkin radiosity [Heckbert and Winget 1991; Zatz 1993] is an exception, but its hierarchical versions have been essentially limited to planar primitives (including wavelet radiosity [Gortler et al. 1993; Schröder 1994]). Thus, with all known global illumination methods, incorporating curved patches is always achieved at the cost of some geometric approximation or important restriction on the shapes allowed: either the input surfaces are tessellated in a number of planar elements until a certain flatness is achieved (which can result in an enormous increase in the number of initial surfaces) or their geometry is approximated for the evaluation of energy transfers (as in the original progressive radiosity solutions – see [Arquès and Michelin 1995] and the references therein). In both cases, the integration of trimmed curved patches is not a natural extension of the polygonal case.

This paper presents the *virtual mesh*, a novel way of computing radiosity on curved surfaces with arbitrary trimming curves that combines the power of adaptive subdivision and functional analysis while avoiding the tessellation of input surfaces. We effectively bridge the gap between general parametric patches and a hierarchical Galerkin formulation with efficient and compact function bases. Furthermore, since our method does not resort to approximations of curved patches, it eliminates almost all geometric sources of error in the algorithm and thus avoids the resolution of a perturbated balance equation [Arvo et al. 1994]. The virtual mesh is thus a powerful solution to the problem of incorporating curved surfaces in radiosity algorithms. With it, scenes made of complex shapes can be illuminated both quickly and very accurately, as Figure 1 illustrates.

The essence of radiosity in a functional setting implies that calculations must be performed over normalized domains (i.e., unit squares or unit right triangles). This is especially important for constructing the basis of functions, defining optimal quadrature rules and maintaining coherence between all the levels in the hierarchy.

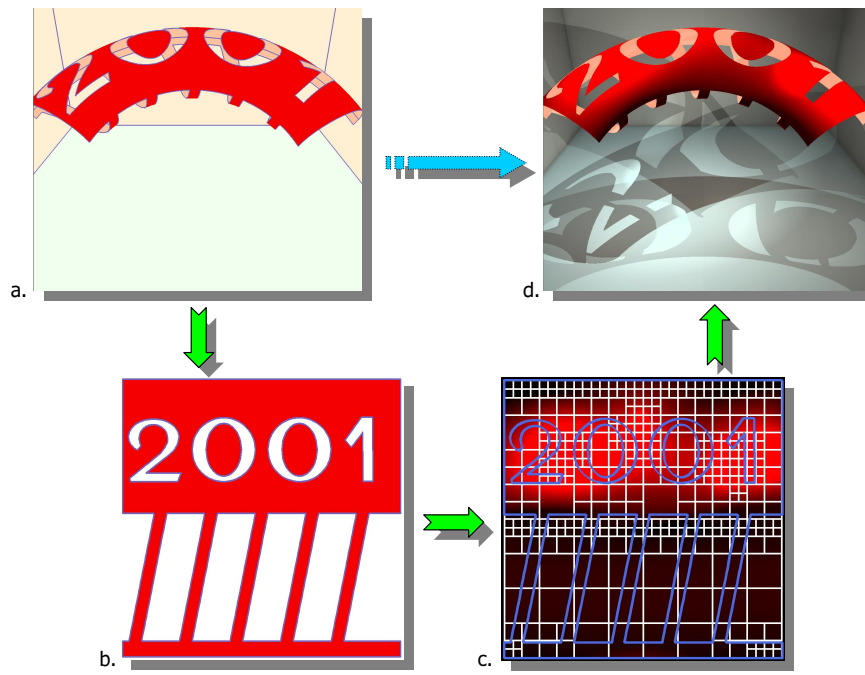

Fig. 2. Overview of the illumination of a trimmed torus patch by four point light sources. a. The input scene. b. The outside of the torus patch mapped to a virtual 2D support. c. The illumination of the 2D support. d. Global illumination of the scene, with radiosity values on the torus deduced from the illumination of its virtual 2D support.

Accordingly, our virtual mesh method uses an abstract representation, made of normalized virtual supports, of the input scene to perform radiosity calculations which would otherwise be intractable. This really is a three-step process (see Figure 2):

1. *map* the input surfaces (curved patches and complex polygons[1]) to normalized planar domains,
2. *extend* the radiosity function to the virtual mesh, the collection of these supports, and compute radiosity using the virtual mesh as the new input geometry,
3. *read off* the intensity values for the original surfaces from the values computed on the virtual mesh.

A key challenge of step 1 is to find appropriate mapping functions, i.e., functions that preserve energy properties of the graphics primitives. Given an input surface, such mappings, called *energy-preserving*, leave the ratio of the areas of two surface elements invariant[2]. The use of energy-preserving mappings allows hierarchical Galerkin methods with high-order basis functions to be used without having to recompute many new attributes at each level of the hierarchy. This invariance

[1]In the rest of this paper, "simple polygons" is used to denote triangles and/or parallelograms and "complex polygons" to denote arbitrary generalized polygons, i.e. possibly non-convex, multiply connected polygons with holes and curved trimming boundaries.
[2]Energy-preserving mappings are also known as *equal-area projections* in cartography.

property has a simple mathematical expression when the input surfaces can be globally parameterized, so we focus on trimmed parametric surfaces in the rest of the paper. This is only a mild restriction since parametric objects are those that can be most easily displayed in computer graphics.

The virtual mesh can be plugged into any advanced variation of radiosity algorithms, including discontinuity meshing, clustering and hierarchical subdivision, and in a variety of finite element methods. To demonstrate its efficiency, we have developed and implemented a high-order wavelet radiosity algorithm that uses our new tool. Since the cost of tessellation increases with the order of the wavelet basis, the virtual mesh is especially applicable to higher-order wavelets. Our implementation can illuminate scenes made of two major types of primitives: *complex polygons* (and in particular ordinary polygons, the most common primitives used by present-day modelers) and *low-degree trimmed algebraic patches*, a primitive set known to encompass 95 percent of conventional, unsculptured parts in industrial environments [Requicha and Voelcker 1982]. It can easily be extended to higher-order surfaces as soon as the appropriate energy-preserving mappings can be numerically evaluated.

The paper is organized as follows. Section 2 describes related work on radiosity with curved primitives. The next two sections give the high-level ideas behind the virtual mesh in radiosity-based global illumination, independently of any resolution method: Section 3 shows how to build the virtual mesh and Section 4 how to use it in a radiosity algorithm. Section 5 then describes how we have incorporated the virtual mesh in a hierarchical radiosity implementation which uses high-order wavelet functions. Section 6 demonstrates the efficiency of the proposed technique on several test cases, before concluding.

2. RELATED WORK

In radiosity algorithms [Cohen and Wallace 1993; Sillion and Puech 1994], one is led to work over very simple supports. This is made necessary because representing functions over general domains is not straightforward. The solution presented in the literature has been to use the tensor product of two 1D bases [Gortler et al. 1993; Zatz 1993]. Often, orthonormal polynomials like the Legendre or Jacobi polynomials are used as 1D bases. Unfortunately, this construction may only be carried out if the support is a simple polygon, i.e., a triangle or a parallelogram. For more general supports, there does not seem to be a simple way of finding 2D basis functions.

In the course of the resolution, one is also led to compute integrals on geometric supports as well as the mean value of the residual. Usually, such computations can only be carried out with numerical methods. Such methods (known as quadrature rules) involve sampling the kernel of the radiosity equation at various points and approximating the integrals as weighted sums of kernel samples. For general supports, there is no automatic way of finding good quadrature points that ensure reconstruction up to a desired accuracy. Also, to avoid recalculation of quadrature points and weights for different types of simple polygons, it is desirable to map the input geometry to normalized domains.

For all these reasons, radiosity-based rendering has largely been limited until now to scenes made of simple polygons. Exceptions to this general rule rely, to illuminate curved surfaces, on some kind of tessellation or geometric approximation [Schäfer

1997; Stamminger et al. 1997] or introduce errors in the resolution by working in the parametric domain without preserving areas [Christensen et al. 1996].

To incorporate curved objects into a hierarchical framework, Schäfer [Schäfer 1997] propose a three-dimensional extension of adaptive meshing, called *object-based meshing*, which avoids the initial subdivision of curved parts. First, each curved object is wrapped in a very crude bounding box. Then, local improvements of the polygonal approximation can be computed at any stage of the algorithm by generating new vertices on the faces of the bounding box and projecting them on the surface of the original object. Since patches at a given level of the hierarchy are no longer the exact unions of child patches and have a different orientation, push-pull operations have to be modified. Also, the influence of geometric approximations on the result may be difficult to evaluate. Stamminger et al. [Stamminger et al. 1997] take a different approach with their *bounded radiosity* method. Their algorithm takes into account the full extent of the interacting objects without resorting to point sampling. The refinement criterion, however, is only a function of upper and lower bounds on form factors, computed using an estimate of the range of normals and boxes bounding the geometry. This may produce unnecessary subdivisions, inducing a lack of efficiency of the hierarchical operations. Note also that this method (as well as the one of Schäfer) assumes the lower-level elements of complex surfaces to be polygons, so it does not work on the exact geometry of the input surfaces.

Christensen et al. [Christensen et al. 1996] present an importance-driven algorithm for the global illumination of glossy environments using piecewise-constant wavelets. A four-dimensional wavelet representation is used to represent the spatially- and angularly-varying radiance distributions across surfaces in the environment. Support for curved surfaces, a feature of the implementation, is limited to surfaces which can be parameterized on the unit square. Handling of parametric surfaces with general trimming curves is thus unclear. Also, only convex patches are allowed, since interactions of an object with itself are not taken into account. In addition, the spatial representation of wavelets on the unit square uses the natural parameters of the patch and so may be heavily biased. Indeed, a large part of the parameter domain may well represent a small part of the patch area, depending on the parameterization of the patch. Since the authors use a uniform sampling of this parameter domain to approximate the transport coefficients, there is no simple way to control the density of sample points needed to sufficiently cover the curved patch and to bound the approximation error. Similar comments apply to the refinement oracle and to the uniform subdivision of the parameter domain, which may corrupt the radiosity computations. In other words, the physical meaning of the quantities computed is lost.

The classical hierarchical radiosity algorithms have a complexity of $O(k^2 + n)$, where k is the number of initial patches and n the number of final patches. Since the $O(k^2)$ term has a strong influence on the resolution process, several algorithms have sought to eliminate it by grouping input surfaces into clusters [Smits et al. 1994; Sillion 1995; Christensen et al. 1997; Willmott et al. 1999] or simplifying the geometry [Rushmeier et al. 1993]. Though they could be used to illuminate curved patches, clustering algorithms have only been used in polygonal scenes up to now. And as indicated by a recent study [Hasenfratz et al. 1999], applying clustering

algorithms to tessellated curved surfaces without preserving topology would be largely inefficient.

To sum up, the best known approaches for globally illuminating curved surfaces suffer from the following shortcomings:

- the traditional meshing approach applied to curved surfaces may lead to severe artifacts and continuity problems. For instance, it can generate a large number of triangles or slivers which are difficult to handle in visibility computations and can cause Z-buffer artifacts when the radiosity solution is visualized;
- all "smart" (i.e., other than straight tessellation) hierarchical radiosity methods for illuminating curved patches introduce errors of a geometrical or physical kind which may deeply affect the accuracy of the radiosity solution;
- since the hierarchical subdivision of curved parts is not the natural extension of the subdivision of flat regions, either geometric values (area, range of normals) have to be recomputed and stored for each patch of the hierarchy [Schäfer 1997], which slows down the simulation, or refinement may be very inefficient (depending on the parameterizations used) because the subdivision of the parameter domain does not correspond to a quadtree of the surface [Christensen et al. 1996];
- all past methods are limited in the range of shapes that they are capable of illuminating.

Our contributions

By contrast, the virtual mesh method has many advantages over these previous attempts at illuminating general parametric surfaces:

- ***genericity*** – It is a natural and strong solution to the rendering of curved objects in a radiosity setting and it provides a currently missing link between hierarchical Galerkin radiosity and parametric surfaces (both convex and concave) with arbitrary trimming curves.
- ***accuracy*** – By keeping a virtual but exact image of the input geometry throughout the resolution process, the virtual mesh avoids most of the geometric errors that plague radiosity algorithms when applied to complex shapes.
- ***flexibility*** – Since the virtual mesh is a "normalized" representation of the input geometry with a set of bijective mappings that allow to go back and forth between real and virtual supports, our method can be combined with any advanced variation of the basic radiosity method.
  To illustrate this flexibility, we present an implementation of the virtual mesh in a hierarchical Galerkin radiosity algorithm. Our radiosity program takes advantage of the very good compression ratios provided by high-order wavelets for storing radiosity. Recent work [Cuny et al. 2000] has shown that high-order wavelets are more efficient and more powerful than piecewise-constant wavelets for the illumination of real-world scenes (using a shooting algorithm and getting rid of links [Stamminger et al. 1998; Winkler 1998]), disproving a common belief.
  The virtual mesh can also be combined with a clustering strategy, to allow for a more efficient manipulation of complex shapes inside clusters and accelerate the computations for the local energy transfers between close surfaces. Motivated by applications where physical accuracy is critical (as in lighting design, where

an exact measure of energy prevails), we have decided not to experiment with clustering in this paper. Indeed, error control is very difficult with clustering and its practical use may result in unexpected behaviors, as recently emphasized in [Hasenfratz et al. 1999].

- ***efficiency*** – Since it gives radiosity algorithms the ability to work with compact representations of real-world scenes, the virtual mesh induces important benefits in terms of computation time, memory consumption and visual quality, compared to the illumination of tessellated objects. In turn, these improvements allow for high-accuracy wavelet radiosity solutions to be computed on large scenes with a reasonable time and memory consumption.

3. BUILDING THE VIRTUAL MESH

As we have illustrated, the main reason why, in the past, radiosity-based rendering has been performed essentially on scenes made of simple polygons is because of the repeated need to do computations on simple normalized domains. Rather than prematurely approximate complex surfaces via triangulation, we use a high-level representation of the scene, the virtual mesh. We show that this representation allows for general planar primitives and trimmed parametric surfaces to be handled as naturally and efficiently as traditional primitives.

This section and the next are dedicated to the principles of the proposed technique, independently of any particular resolution algorithm of the radiosity equation. We start here by giving a thorough presentation of the construction of the virtual mesh for parametric patches. Section 4 will then show how this representation can be used to efficiently and accurately compute radiosity on trimmed curved patches and complex polygons.

The main idea of the virtual mesh is to compute radiosity on a set of unit virtual supports representing the (possibly curved) input surfaces. For this to have a physical meaning, the mappings involved must be energy-preserving. (As we shall see, this invariance property is crucial when using non-constant basis functions in a hierarchical Galerkin setting – Section 5 – and allows for important speedups – Section 6). In mathematical terms, since radiosity is a measure of power by unit area, a mapping is *energy-preserving* if it leaves the ratio of the areas of two surface elements invariant, i.e., if its Jacobian[3] is constant.

Now, a direct mapping of a curved or complex polygonal primitive S onto a unit square is not always possible, so we separate the problem at hand into three different tasks (see Figure 3):

(1) find a constant Jacobian mapping Ψ^{-1} that maps a curved patch S to the plane (Section 3.1),

(2) enclose the image S' of the patch with a parallelogram $\bar{S}'$ (Section 3.2),

(3) map the parallelogram $\bar{S}'$ onto a unit square $\tilde{S}$ (Section 3.3).

Obviously, if S is already planar, the first step is skipped.

[3]See [Kaplan 1984, pp. 98–99 and 238–245] for precise mathematical definitions and properties.

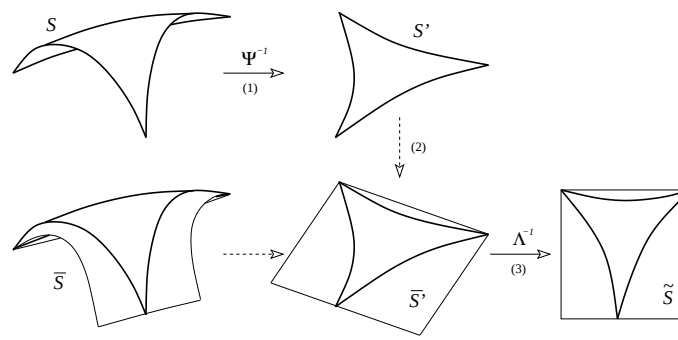

Fig. 3. Mapping a trimmed curved patch to a unit square domain.

3.1 Mapping curved patches to the plane

Finding a mapping with a constant Jacobian which "flattens" a curved surface is a key issue in our method. It allows calculations to be performed on a simpler geometry while preserving energy properties of the input patch. This section examines this issue.

3.1.1 *Intuitively.* Let S be a smooth curved patch parameterized by

$$\mathbf{x}(\theta,\varphi) = (x(\theta,\varphi), y(\theta,\varphi), z(\theta,\varphi)), \quad (\theta,\varphi) \in I, I \subseteq \mathbb{R}^2.$$

θ and φ are called the *natural parameters* of S.

In what follows, subscripts indicate partial derivatives. The isoparameter curves

$$C^{\theta} : \mathbf{x}(\theta = \text{cst}, \varphi), \qquad C^{\varphi} : \mathbf{x}(\theta, \varphi = \text{cst}),$$

where cst means constant, define a net of curves on S. At each parameter pair $(\theta_0, \varphi_0) \in I$, the vector $\mathbf{x}_\theta(\theta_0, \varphi_0)$ is tangent to C^{θ_0} at $\mathbf{x}_0 = \mathbf{x}(\theta_0, \varphi_0)$ and the speed of the position vector along C^{θ_0} at $\mathbf{x}_0$ is $\|\mathbf{x}_\theta(\theta_0, \varphi_0)\|$. Similar observations hold for C^{φ_0}.

The quantity $\|\mathbf{x}_\theta(\theta_0, \varphi_0) \times \mathbf{x}_\varphi(\theta_0, \varphi_0)\|\, d\theta\, d\varphi$ measures the area of a differential element around point $\mathbf{x}_0$. The image of this differential element by the "natural coordinates" mapping

$$\chi^{-1} : (x, y, z) \longmapsto (\theta, \varphi)$$

is a rectangle of area $d\theta\, d\varphi$. Thus χ is energy-preserving if

$$\|\mathbf{x}_\theta(\theta_0, \varphi_0) \times \mathbf{x}_\varphi(\theta_0, \varphi_0)\| = \text{cst}, \quad \forall (\theta_0, \varphi_0) \in I.$$

There is no reason for this to be the case. Thus, to find an energy-preserving mapping, we need to reparameterize S by the parameters (u, v) in such a way that

$$\|\mathbf{x}_u(\theta_0, \varphi_0) \times \mathbf{x}_v(\theta_0, \varphi_0)\| = \text{cst}, \quad \forall (\theta_0, \varphi_0) \in I. \tag{1}$$

It is always possible to construct an energy-preserving mapping $\Psi : (u, v) \mapsto (x, y, z)$ by taking for instance $u = \theta$ and adjusting the speed of the position vector along C^{φ} in such a way that the constraint of Equation (1) is satisfied at each point of S.

3.1.2 *Formally.* What the above shows is that there exists at least one energy-preserving mapping that sends S to a suitable parameter domain. Now, there are

actually many qualitatively different energy-preserving mappings for a given curved surface S and some may be easier to deal with computationally than others.

The general case is to consider $\theta = f(u, v)$ and $\varphi = g(u, v)$. Actually, we have observed that it is usually not necessary to have both u and v depend on θ and φ. So assume for now that $\theta = f(u), \varphi = g(u, v)$. Let us determine the constraint that f and g must satisfy for Ψ to be energy-preserving. Since

$$\begin{cases} \mathbf{x}_u(\theta, \varphi) = f'(u)\mathbf{x}_\theta(\theta, \varphi) + g_u(u, v)\mathbf{x}_\varphi(\theta, \varphi), \\ \mathbf{x}_v(\theta, \varphi) = g_v(u, v)\mathbf{x}_\varphi(\theta, \varphi), \end{cases}$$

the Jacobian of Ψ is [Kaplan 1984]:

$$\mathcal{J}_\Psi(u, v) = \|\mathbf{x}_u(\theta, \varphi) \times \mathbf{x}_v(\theta, \varphi)\| = |f'(u)g_v(u, v)| \, \|\mathbf{x}_\theta(\theta, \varphi) \times \mathbf{x}_\varphi(\theta, \varphi)\|.$$

Now, Ψ and Ψ^{-1} are energy-preserving if the Jacobian $\mathcal{J}_\Psi$ is constant, say equal to $1/c, c \neq 0$. In virtue of the theorem on the derivatives of inverse functions, we get (up to sign)

$$\frac{\partial v}{\partial \varphi}(\theta, \varphi) = \frac{1}{g_v(u, v)} = c|f'(u)| \, \|\mathbf{x}_\theta(\theta, \varphi) \times \mathbf{x}_\varphi(\theta, \varphi)\|.$$

Integrating with respect to φ and replacing θ and φ by $f(u)$ and $g(u, v)$ respectively leads to

$$v = c|f'(u)| \int_0^{g(u,v)} \|\mathbf{x}_\theta(f(u), t) \times \mathbf{x}_\varphi(f(u), t)\| \, dt. \tag{2}$$

Any choice of f and g satisfying the constraint of Equation (2) gives an energy-preserving mapping $\Psi^{-1} : (x, y, z) \mapsto (u, v)$ and its inverse.

For the different operations we need to carry out (computing the virtual mesh from the input geometry, forwarding visibility queries to the virtual mesh, reconstructing the radiosity function on curved patches), we want to be able to go back and forth between $(x, y, z), (\theta, \varphi)$ and (u, v). Thus, we need to know the inverse of all the different mappings involved. This is not necessarily as difficult as it may seem. Indeed, all these mappings are bijective (except possibly on a set of measure zero) so that, in case the inversion cannot be achieved exactly, a numerical algorithm will do a fine job with arbitrary precision. Also, the parameterizations we shall consider in this paper all involve circular or hyperbolic functions, for which the inverse mapping $(x, y, z) \mapsto (\theta, \varphi)$ can be computed exactly.

3.1.3 *Example: a ring torus.* Consider for instance the doughnut-like shape of Figure 4.a. Mathematically, it corresponds to a patch on the ring torus given by the parameterization

$$\begin{cases} x(\theta, \varphi) = \cos\theta\,(\cos\varphi + 2), \\ y(\theta, \varphi) = \sin\theta\,(\cos\varphi + 2), \\ z(\theta, \varphi) = \sin\varphi, \end{cases} \quad \theta \in \left[0, \frac{\pi}{2}\right), \varphi \in [0, \pi).$$

Equation (2) rewrites as:

$$v = c|f'(u)|\Big(g(u, v) + \frac{1}{2}\sin g(u, v)\Big). \tag{3}$$

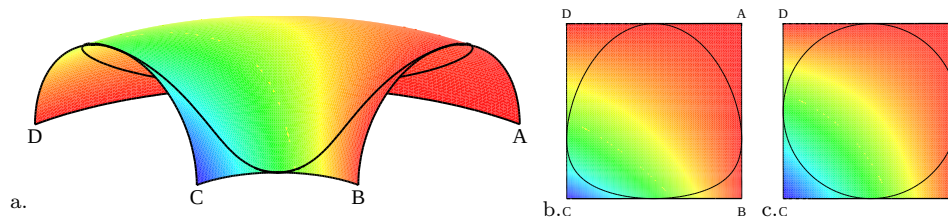

Fig. 4. Mapping a torus patch with a curve drawn on it to the domain $[0, \pi]^2$. a. A ring torus patch. b. A mapping of this patch with constant Jacobian. c. A mapping with non-constant Jacobian.

Clearly, choosing f to be a linear function of u will get rid of the $|f'(u)|$ term. Let us for instance choose $f(u) = u/2$ and $c = 1$. Equation (3) boils down to a constraint on g for $\Psi : (u, v) \mapsto (x, y, z)$ to be energy-preserving:

$$v = g(u, v) + \frac{1}{2} \sin\left(g(u, v)\right).$$

Here, it turns out that g does not depend on u, so we may write $g(u, v) = g(v)$. With the above equation, g can be determined numerically, from which we can deduce the energy-preserving mapping Ψ:

$$\Psi : (u, v) \longmapsto \Big(\cos\frac{u}{2}(\cos g(v) + 2), \sin\frac{u}{2}(\cos g(v) + 2), \sin g(v) \Big).$$

Since $v = g(v) + \frac{1}{2} \sin g(v) = \varphi + \frac{1}{2} \sin\varphi$, the inverse mapping Ψ^{-1} is:

$$\Psi^{-1} : (x, y, z) \longmapsto \Big(2\theta, \varphi + \frac{1}{2} \sin\varphi \Big).$$

The result of applying Ψ^{-1} to our ring torus patch gives the planar surface displayed in Figure 4.b. This should be compared with Figure 4.c which shows the result of applying the mapping $(x, y, z) \mapsto (2\theta, \varphi)$, the Jacobian of which is non-constant, which sends a 3D point to its natural coordinates (up to a constant factor).

Appendix A gives additional examples of energy-preserving mappings for a few curved surfaces of interest.

3.2 Enclosing complex polygonal primitives

What the previous section has essentially proved is that a curved input patch S can be replaced with a planar primitive S'. Now, this primitive may be somewhat arbitrary (as in Figure 3): it may have curved boundaries, holes... In general, it is not possible to map a curved patch directly onto a unit square in such a way that the mapping is bijective, except in important but special cases (an example of which is the torus patch of Figure 4). An intermediate step is to bound the planar primitive S' by a parallelogram $\bar{S}'$. We do it in such a way as to minimize the voids (i.e., the area of $\bar{S}' \setminus S'$). This way, we also minimize the regions where the radiosity function has to be extended (see Section 4.1) and thus the amount of unnecessary calculations.

How does one get a good bounding box? If S' has a polygonal outer boundary with n vertices, we start by computing its convex hull in $O(n)$. Then we use the

minimal enclosing parallelogram algorithm of Schwarz et al. [Schwarz et al. 1995] which has a complexity linear in the number of vertices. The resulting enclosing parallelogram $\bar{S}'$ indeed has the property that it minimizes the voids. If S' has a curved boundary, we simply discretize this boundary, compute a minimal enclosing parallelogram of the discrete curve with the algorithm of Schwarz et al. and use this parallelogram as the bounding box of S'. Note that the overhead of computing an enclosing parallelogram for each input patch – something that has to be done only once at the beginning of the algorithm – is negligible compared to the overall computation time (i.e., less than one thousandth of the total).

3.3 Mapping quadrilaterals to unit squares

The last step of the chain is to map the parallelogram $\bar{S}'$ found previously onto a unit square $\tilde{S}$. This is an easy task, which can be done using an affine mapping Λ^{-1} whose inverse can be written as follows:

$$(u', v') = \Lambda(\tilde{u}, \tilde{v}) = A + \tilde{u}\overrightarrow{AB} + \tilde{v}\overrightarrow{AD},$$

where the points A, B, C, D are the vertices of the quadrilateral. As can readily be observed, the Jacobian of this mapping is constant: it is equal to the area of $\bar{S}'$.

4. USING THE VIRTUAL MESH

Now that we have shown how to build the virtual mesh, our aim in this section is to discuss its use in a radiosity algorithm. The main issue here is that, generally speaking, not all points of the unit virtual support $\tilde{S}$ correspond to points of the original S. Thus, if we want to compute radiosity on $\tilde{S}$, we need to extend the definition of the radiosity function on S to $\tilde{S}$ (or rather to $\bar{S}$, the 3D equivalent of $\tilde{S}$).

Section 4.1 presents our simple extension of the radiosity function to the virtual mesh. Then, Section 4.2 shows how to compute radiosity on the virtual mesh, knowing that it represents a geometry that is larger than the original geometry. Once radiosity values have been computed on the virtual mesh up to the desired convergence rate, they can be lifted to the original mesh using the inverse mappings $\tilde{S} \rightarrow \bar{S}' \rightarrow S$ and subsequently passed to the rendering engine as in traditional radiosity algorithms.

4.1 Defining radiosity on $\tilde{S}$

In general, there are "more" points on $\tilde{S}$ than on S. We examine here how radiosity can be defined on those extra points.

Let $\bar{S}$ be the pre-image of $\tilde{S}$ by $(\Psi \circ \Lambda)^{-1}$ (see Figure 3). Call a point of $\tilde{S}$ *real* if it has a pre-image on S, *virtual* otherwise (i.e., if belongs to $\bar{S} \setminus S$). The extension of the radiosity function to virtual supports should follow two basic rules:

- The extended function is the same as the original one on real points of $\tilde{S}$.
- The extended function should be as simple and continuous as possible on virtual points of $\tilde{S}$ and on the boundaries between real and virtual points.

The first rule is essential since we want to deduce radiosity values on S from radiosity values on $\tilde{S}$. The second rule is no less important. Indeed, in hierarchical radiosity (Section 5), the radiosity value at any point of $\tilde{S}$ (e.g., at a virtual point) can

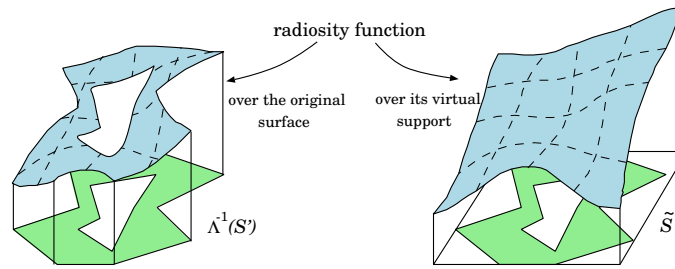

Fig. 5. Smoothly extending the radiosity function.

influence the radiosity value at any other point of $\tilde{S}$ (e.g., at a real point). Thus, the radiosity function must be computed with the same precision at all points of $\tilde{S}$.

Since discontinuities of the derivatives of the radiosity function are only due to discontinuities of the visibility function [Heckbert 1992], we do not want our extension to introduce new visibility discontinuities. To this aim, we introduce an extension $\tilde{v}$ of the visibility function v as follows. Visibility between a point $\mathbf{p}$ of the environment and a point $\bar{\mathbf{q}}$ of $\bar{S}$ is defined as the visibility between $\mathbf{p}$ and $\mathbf{q}$, where $\mathbf{q}$ is the point of S that is closest to $\bar{\mathbf{q}}$:

$$\tilde{v}(\mathbf{p}, \bar{\mathbf{q}}) = v(\mathbf{p}, \mathbf{q}), \qquad \|\mathbf{q} - \bar{\mathbf{q}}\| = \min_{\mathbf{q}' \in S} \|\mathbf{q}' - \bar{\mathbf{q}}\|.$$

The radiosity function over $\tilde{S}$ is then defined in a classical way, but using $\tilde{v}$ as visibility function (see Figure 5):

$$B(\mathbf{x}) = E(\mathbf{x}) + \rho(\mathbf{x}) \int_{\bar{S}} B(\mathbf{y}) G(\mathbf{x}, \mathbf{y}) \tilde{v}(\mathbf{x}, \mathbf{y}) \, dA.$$

Even though this may not be the best extension of the radiosity function, it is very simple and has proved to be very reliable in practice.

4.2 Computing radiosity on $\tilde{S}$

Now that we have properly defined the radiosity function at all points of the virtual mesh, we are almost ready to compute the extended radiosity function and we must do so in a meaningful way. The general idea is to work with our high-level representation whenever possible and go back to the original geometry only when needed (i.e., for visibility queries). The main issue here is that since we work with a virtual geometry that represents more points than the original geometry, we must be careful not to propagate the wrong quantity of energy.

Let us analyze the three different cases, i.e., when S is a receiver, a blocker or an emitter.

S is a receiver. When S acts as a receiver, we use $\tilde{S}$ exactly as if this virtual support was the receiver. Real and virtual points are handled in exactly the same way and with the same accuracy. This way, the radiosity function is consistent over the whole virtual support and can be used in a hierarchical algorithm without the virtual points disturbing the real points of $\tilde{S}$.

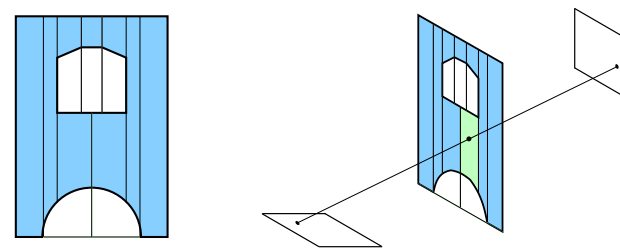

Fig. 6. Trapezoidal map of an arrangement of line segments and curved arcs.

S is a blocker. When S acts as a blocker, we must use its real geometry, and not $\tilde{S}$, in visibility calculations. We proceed as follows.

Let $\dot{S}$ be the untrimmed equivalent of S (i.e., if S is defined by $\mathbf{x}(\theta,\varphi), (\theta,\varphi) \in I$, the surface $\dot{S}$ is defined by $\mathbf{x}(\theta,\varphi), (\theta,\varphi) \in \mathbb{R}^2$). First, we compute the intersection of the visibility ray with $\dot{S}$:

- If S is curved, computing the intersection is not too hard since the surface is given in parametric form. If there is an intersection, we project it using $(\Psi \circ \Lambda)^{-1}$. If the projection falls outside of the virtual support $\tilde{S}$, we are done since this means that the 3D intersection point is not on S.
- If S is planar, then $\dot{S}$ is a plane so computing the intersection point (there is almost always one) is easy. Project this point using Λ^{-1}. As before, if the projection falls outside of the virtual support $\tilde{S}$, we are done.

To determine if the projected point is a real or a virtual point of $\tilde{S}$ (i.e., if S is a blocker for the ray considered or not), we use a trapezoidal map algorithm [Boissonnat and Yvinec 1998] which we have slightly improved to take curved arcs into account. The rough idea is to project the trimming curves of S (the projected curves come in two flavors: line segments and curved arcs) and to divide the map area (here: the virtual support $\tilde{S}$) into a number of "slabs" by drawing vertical lines through the ends of each line segment and through the inflections and extrema in the u and v directions of each curved arc. It is then possible with a fast binary search to find which slab the projected point is in and consequently whether S is a blocker for the current visibility ray (see the example of Figure 6).

S is an emitter. When S acts as an emitter, we need to be careful because both real and virtual points of its virtual support have received energy. In other words, since only the real points have a physical meaning, $\tilde{S}$ has accumulated more energy than it should have. To avoid repropagating this extra energy, the idea is to continue working with the extended radiosity function on $\tilde{S}$ but to use as weights of quadrature points only a fraction of what they are over real regions, as in Figure 7. (Adjusting quadrature weights to the virtual support is similar in spirit to the shadow masks used by Zatz to smooth the shadow discontinuities out of the radiosity distribution seen by the Galerkin method [Zatz 1993].)

To compute the interaction between two basis functions (one on the emitter and one on the receiver) using a quadrature rule, we are lead to compute the interaction between a function on S and a point on the receiver. Assume for now that the whole of $\tilde{S}$ shoots energy. We need to evaluate integrals of the form $\int_{\text{emitter}} f(\mathbf{p})\, dA$, where

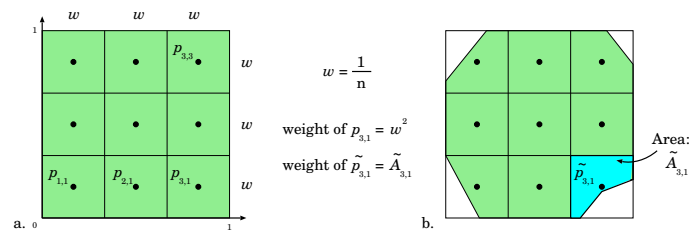

Fig. 7. Example of adjusting quadrature weights to virtual supports when S is considered an emitter. a. Classical 2D quadrature points and weights. b. Modified weights on virtual supports.

dA is an element of area on $\tilde{S}$, which are approximated as weighted sums

$$\sum_{\mathbf{p}_i \in \text{ emitter}} w_i f(\mathbf{p}_i),$$

where w_i is the weight associated to quadrature point $\mathbf{p}_i$. Since we work with constant Jacobian mappings, a possibility is to place a regular $n \times n$ grid on $\tilde{S}$, take the quadrature points at the center of each cell and compute

$$\int_S f(\mathbf{p})\, dA \approx \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} f(\mathbf{p}_{i,j}),$$

where $\mathbf{p}_{i,j} = \left(\frac{2i-1}{2n}, \frac{2j-1}{2n}\right)$ and $w_{i,j} = \frac{1}{n^2}$ is the area of the cell in which $\mathbf{p}_{i,j}$ falls.

Now, a simple way to adjust this quadrature rule to the real part of the virtual support and to avoid energy overpropagation is to take as quadrature points $\tilde{\mathbf{p}}_{i,j} = \mathbf{p}_{i,j}$ and to use as quadrature weight of $\tilde{\mathbf{p}}_{i,j}$ the area of the "real" part of the cell in which it falls, i.e. the part that corresponds to real points of S (as in Figure 7.b). Clearly, as the size n of the grid is increased, the total area of the mixed cells (i.e., cells having both real and virtual points) will decrease as a function of $\frac{1}{n^2}$, so that the right amount of energy will be emitted.

5. VIRTUAL MESH IN HIGH-ORDER WAVELET RADIOSITY

Since the virtual mesh uncouples the input geometry from the representation of radiosity, the general method presented in the previous section can be efficiently combined with any advanced radiosity algorithm. In particular, it fits nicely in a hierarchical Galerkin radiosity algorithm. We report here an implementation of a high-order wavelet radiosity algorithm with the virtual mesh.

5.1 Algorithm overview

Our hierarchical radiosity algorithm paired with the virtual mesh is outlined in Algorithm 1. It is based on the well-known shooting method with Southwell relaxation. As can readily be seen, this algorithm has a lot in common with a classical hierarchical radiosity algorithm, apart from some steps which we shall detail in the following sections.

The interested reader is referred to [Cuny et al. 2000] for the details of our implementation. Suffice it to say for the purpose of the present paper that the refinement oracle is similar to the one of [Bekaert and Willems 1996] (estimate

Algorithm 1 Overview of HR with the virtual mesh.

for all $i \in \{1, \ldots, n\}$ **do**
  $\tilde{S}_i \longleftarrow \mathit{VirtualSupport}(S_i)$
  $\tilde{B}_i \longleftarrow \tilde{E}_i, \quad \Delta\tilde{B}_i \longleftarrow \tilde{E}_i$
end for
while $\sum_k \int_{S_k,\lambda} \Delta\tilde{B}_k > \varepsilon$ **do**
  Choose i such that $\tilde{S}_i$ has max unshot energy $\Delta\tilde{B}_i$
  $\mathit{Push/Pull\text{-}Residuals}(\tilde{S}_i)$
  for all $j \in \{1, \ldots, n\}$ **do**
    $\tilde{\mathcal{E}}_j \longleftarrow \mathit{HR\text{-}Transfer}(S_i \longrightarrow \tilde{S}_j, \Delta\tilde{B}_i)$
    $\tilde{B}_j \longleftarrow \tilde{B}_j + \tilde{\mathcal{E}}_j, \quad \Delta\tilde{B}_j \longleftarrow \Delta\tilde{B}_j + \tilde{\mathcal{E}}_j$
  end for
  $\Delta\tilde{B}_i \longleftarrow 0$
end while
for all $i \in \{1, \ldots, n\}$ **do**
  $\mathit{UpdateHierarchy}(\tilde{S}_i)$
  $\mathit{Draw}(S_i)$
end for

errors on the radiosity function using control points on the receiver) and that we use shooting rather than the more conventional gathering approach because it is what allows to get rid of links (as explained in [Stamminger et al. 1998] and [Winkler 1998]) and also because it is what makes higher-order wavelets good and efficient for radiosity.

We use the most general wavelet basis functions known in the radiosity field, i.e. high-order wavelets. We have observed that for linkless radiosity, and with the refinement oracle described in [Cuny et al. 2000], a previously noted annoying behavior of high-order tree-like wavelets (namely, they can produce ringing discontinuities when the refinement oracle is not high enough [Willmott and Heckbert 1997]) is rarely a problem.

When working over curved surfaces, which may exhibit large variations of curvatures and normals, it is desirable to increase the order of the wavelet functions to better capture the details of the radiosity function and reduce the number of final patches. For instance, we have determined that when the input scene is made of quadric surface patches, the best visual results are obtained using linear or quadratic wavelets. And as shown in [Cuny et al. 2000], using linear ($\mathcal{M}_2$[4]) or quadratic ($\mathcal{M}_3$) wavelets in a radiosity algorithm is actually faster and less memory-consuming than using Haar ($\mathcal{M}_1$) wavelets, for the specific case of a shooting algorithm that does not store links.

As said in Section 4, we use the original geometry of the surface only when needed, i.e., for visibility queries. This means that the hierarchy of subdivisions of the surface is constructed over the virtual mesh and not over the original surface. In Section 5.2, we show that the constant Jacobian property is crucial for the push-pull

[4]The notation $\mathcal{M}_n$ refers to the basis of multi-wavelets of order n, i.e., the wavelets whose smoothing functions are tensor products of the first n Legendre polynomials [Gortler et al. 1993].

coefficients computed on the virtual mesh to be meaningful for the original surface and provides a very natural answer to the extension of wavelet radiosity to curved surfaces and complex polygons.

Two points will then remain to be discussed, which correspond to steps underlined in red in Algorithm 1. Section 5.3 shows how to handle self-shooting surfaces (line 8 of the algorithm, a surface can interact with itself) and Section 5.4 details the management of the hierarchy (line 9).

5.2 Push-pull coefficients

In this section, we prove that the use of energy-preserving mappings greatly simplifies the computation of push-pull coefficients over general supports and allows for a key property of classical push-pull coefficients (namely, their independence on the level of the hierarchy) to be transfered to general parametric surfaces.

To study this property, recall that 1D wavelet functions are generated from a single function ϕ by dilations and translations [Schröder 1994]:

$$\phi_i^m(x) = \sqrt{2^m}\phi(2^m x - i).$$

Let S_i^m be the support of ϕ_i^m. Let $\tilde{f}$ be the projection of a univariate function f defined on $[0,1]$ in a tree wavelet basis. Then we have:

$$\tilde{f}^m(x) = \sum_i \alpha_i^m \phi_i^m(x),$$

where m is the level of representation and the α_i^m are weights associated to ϕ_i^m. The hierarchical properties of wavelets allow information to be transfered from one level of the hierarchy to the next or the previous by computing a set of *push-pull coefficients* $h_{i,j}^m$ and $g_{i,j}^m$:

$$\alpha_j^{m+1} = \sum_i g_{i,j}^m \, \alpha_i^m, \quad \alpha_i^{m-1} = \sum_j h_{i,j}^m \, \alpha_j^m. \tag{4}$$

The classical case. Suppose given $\tilde{f}^m$ and α_i^m. Since level $m+1$ is finer than level m, $\tilde{f}^m$ can also be represented at level $m+1$. In other words, there exist coefficients α_i^{m+1} such that

$$\sum \alpha_i^m \phi_i^m(x) = \sum \alpha_i^{m+1} \phi_i^{m+1}(x).$$

Multiplying both sides by $\phi_j^{m+1}(x)$ and taking the inner product on the space of functions yields:

$$\sum \alpha_i^m \left\langle \phi_i^m, \phi_j^{m+1} \right\rangle_{S_j^{m+1}} = \sum \alpha_i^{m+1} \left\langle \phi_i^{m+1}, \phi_j^{m+1} \right\rangle_{S_j^{m+1}}. \tag{5}$$

The classical wavelet functions form an orthonormal basis. Thus:

$$\left\langle \phi_i^{m+1}, \phi_j^{m+1} \right\rangle_{S_j^{m+1}} = \delta_{ij},$$

where δ_{ij} is the Kronecker delta. We end up with

$$\alpha_j^{m+1} = \sum \alpha_i^m \left\langle \phi_i^m, \phi_j^{m+1} \right\rangle_{S_j^{m+1}}.$$

With this and another similar calculation, we conclude that:

$$g^m_{i,j} = \left\langle \phi^m_i, \phi^{m+1}_j \right\rangle_{S^{m+1}_j}, \quad h^m_{i,j} = \left\langle \phi^m_i, \phi^{m+1}_j \right\rangle_{S^m_i}.$$

Since classical wavelet functions are defined on normalized domains, one can prove that the push-pull coefficients do not depend on the level of representation m. For instance, for $h^m_{i,j}$:

$$\begin{aligned} h^m_{i,j} = \int_{S^m_i} \phi^m_i(x)\phi^{m+1}_j(x)\,dx &= 2^m\sqrt{2}\int_{S^m_i} \phi(2^m x - i)\phi(2^{m+1}x - j)\,dx, \\ &= \sqrt{2}\int_{[0,1]} \phi(y)\phi(2y + 2i - j)\,dy, \qquad (y = 2^m x - i), \\ &= \left\langle \phi, \phi^1_{2i-j} \right\rangle_{[0,1]} = h^0_{0,2i-j}. \end{aligned}$$

Similar properties can be proved for 2D wavelet functions. Suppose that these functions are defined as tensor products of 1D wavelet functions (other definitions are possible, but lead to the same conclusions):

$$\phi^m_{i,j}(x, y) = \phi^m_i(x)\phi^m_j(y).$$

A scalar function f defined over $[0, 1]^2$ (e.g., the radiosity function) can be projected in this basis of functions:

$$\tilde{f}(x, y) = \sum_{i,j} \alpha^m_{i,j}\phi^m_{i,j}(x, y).$$

As before, a set of push-pull coefficients can be defined which allow for information to be transfered between levels of the hierarchy:

$$h^m_{(i,j),(k,l)} = \left\langle \phi^m_{i,j}, \phi^{m+1}_{k,l} \right\rangle_{S^m_{i,j}}, \qquad g^m_{(i,j),(k,l)} = \left\langle \phi^m_{i,j}, \phi^{m+1}_{k,l} \right\rangle_{S^{m+1}_{k,l}}.$$

And as before, these coefficients do not depend on m. For instance:

$$\begin{aligned} h^m_{(i,j),(k,l)} &= \int_{S^m_{i,j}} \phi^m_{i,j}(x, y)\phi^{m+1}_{k,l}(x, y)\,dx\,dy, \\ &= \Big(\int_{S^m_i} \phi^m_i(x)\phi^{m+1}_k(x)\,dx\Big)\Big(\int_{S^m_j} \phi^m_j(x)\phi^{m+1}_l(x)\,dy\Big), \\ &= h^m_{i,k}h^m_{j,l} = h^0_{0,2i-k}h^0_{0,2j-l} = h^0_{(0,0),(2i-k,2j-l)}. \end{aligned}$$

The general case. The above properties are very desirable to avoid heavy and costly calculations each time a hierarchy is created. We now show that to have similar properties in the general case, we need to work with constant Jacobian mappings.

When the support is not normalized (say, in 1D, a curved arc), the basis functions $\hat{\phi}^m_i$ are not orthonormal and they are different from the classical wavelet functions. In this case, combining Equations (4) and (5) gives a definition of the push-pull coefficients in matrix form:

$$\mathcal{H}^m = (\mathcal{A}^m)^{-1}\mathcal{B}^m, \qquad \mathcal{G}^m = (\mathcal{C}^m)^{-1}\mathcal{D}^m,$$

where

$$\mathcal{H}^m = \left(\hat{h}^m_{i,j}\right), \mathcal{G} = \left(\hat{g}^m_{i,j}\right), \quad \mathcal{A}^m = \left(\left\langle \hat{\phi}^m_i, \hat{\phi}^m_j \right\rangle_{\hat{S}^m_i}\right), \mathcal{C}^m = \left(\left\langle \hat{\phi}^m_i, \hat{\phi}^m_j \right\rangle_{\hat{S}^m_j}\right),$$
$$\mathcal{B}^m = \left(\left\langle \hat{\phi}^m_i, \hat{\phi}^{m+1}_j \right\rangle_{\hat{S}^m_i}\right), \mathcal{D}^m = \left(\left\langle \hat{\phi}^m_i, \hat{\phi}^{m+1}_j \right\rangle_{\hat{S}^{m+1}_j}\right).$$

We have a similar definition in the 2D case which involves coefficients of the form:

$$\left\langle \hat{\phi}^m_{i,j}, \hat{\phi}^m_{k,l} \right\rangle_{\hat{S}^m_{i,j}}, \quad \left\langle \hat{\phi}^m_{i,j}, \hat{\phi}^m_{k,l} \right\rangle_{\hat{S}^m_{k,l}}, \quad \left\langle \hat{\phi}^m_{i,j}, \hat{\phi}^{m+1}_{k,l} \right\rangle_{\hat{S}^m_{i,j}}, \quad \left\langle \hat{\phi}^m_{i,j}, \hat{\phi}^{m+1}_{k,l} \right\rangle_{\hat{S}^{m+1}_{k,l}}.$$

Let S be a parametric surface. Recall the notations of Section 3:

$$S \xrightarrow{\Psi^{-1}} S', \quad \bar{S}' \xrightarrow{\Lambda^{-1}} \tilde{S},$$

where $\Psi^{-1} : (x, y, z) \mapsto (u, v)$ sends S to the parametric domain (u, v), $\bar{S}'$ is a parallelogram enclosing the image S' of S by Ψ^{-1} and $\Lambda^{-1} : (u', v') \mapsto (\tilde{u}, \tilde{v})$ sends $\bar{S}'$ to $[0, 1]^2$. Let $\hat{S}^m_{i,j} = (\Psi \circ \Lambda)(S^m_{i,j})$. We aim to see how push-pull coefficients on S, or more precisely on $\bar{S} = (\Psi \circ \Lambda)(\tilde{S})$, relate to those of $\tilde{S}$. We start by computing the following coefficient:

$$\begin{aligned}
\left\langle \hat{\phi}^m_{i,j}, \hat{\phi}^{m+1}_{k,l} \right\rangle_{\hat{S}^m_{i,j}} &= \int_{\hat{S}^m_{i,j}} \hat{\phi}^m_{i,j}(\theta, \varphi) \hat{\phi}^{m+1}_{k,l}(\theta, \varphi)\, dA, \\
&= \int_{\check{S}^m_{i,j}} \check{\phi}^m_{i,j}(u, v) \check{\phi}^{m+1}_{k,l}(u, v) \mathcal{J}_\Psi(u, v)\, du\, dv, \\
&= \int_{S^m_{i,j}} \phi^m_{i,j}(\tilde{u}, \tilde{v}) \phi^{m+1}_{k,l}(\tilde{u}, \tilde{v}) \mathcal{J}_\Psi(u, v) \mathcal{J}_{\Lambda|_{S^m_{i,j}}}\, d\tilde{u}\, d\tilde{v},
\end{aligned}$$

where $\Lambda|_{S^m_{i,j}}$ is the restriction of Λ to $S^m_{i,j}$. As seen in Section 3.3, $\mathcal{J}_\Lambda$ is a constant function of $\tilde{u}, \tilde{v}$ so it can be taken out of the integral ($\mathcal{J}_{\Lambda|_{S^m_{i,j}}}$ depends on m though, since it is related to the area of $S^m_{i,j}$). Now, if Ψ is energy-preserving, $\mathcal{J}_\Psi$ is constant and can be taken out of the integral. In this case,

$$\left\langle \hat{\phi}^m_{i,j}, \hat{\phi}^{m+1}_{k,l} \right\rangle_{\hat{S}^m_{i,j}} = \mathcal{J}_\Psi \mathcal{J}_{\Lambda|_{S^m_{i,j}}} \int_{S^m_{i,j}} \phi^m_{i,j}(u, v) \phi^{m+1}_{k,l}(u, v)\, du\, dv = \mathcal{J}_\Psi \mathcal{J}_{\Lambda|_{S^m_{i,j}}} h^m_{(i,j),(k,l)}.$$

Similarly,

$$\left\langle \hat{\phi}^m_{i,j}, \hat{\phi}^m_{k,l} \right\rangle_{\hat{S}^m_{i,j}} = \mathcal{J}_\Psi \mathcal{J}_{\Lambda|_{S^m_{i,j}}} \delta_{i,k} \delta_{j,l}.$$

This means that the formula defining the push-pull coefficients reduces in the energy-preserving case to a single term, $\hat{h}^m_{(i,j),(k,l)} = h^m_{(i,j),(k,l)}$, which we know to be independent of m.

Wrapping up. What we just proved is an important result. First, it gives yet another hint that the virtual mesh with constant Jacobian mappings is the natural way of extending traditional radiosity techniques to complex primitives. Second, it means that computing push-pull coefficients on a parametric surface is very simple when using energy-preserving projections, while they involve a double sum of coefficients in the general case. Third, and most importantly, the push-pull coefficients $\hat{h}^m_{(i,j),(k,l)}$ do not depend on m. They can thus be precomputed and stored at

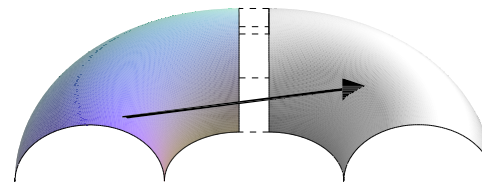

Fig. 8. A self-shooting surface shoots on a decoy.

the top level of the hierarchy, which simplifies the process of redistributing energy within the hierarchy. We have determined that this speeds up the process by a factor of 3 for linear wavelets and by a factor of 7 for quadratic wavelets. The explanation is simple: the number of basis functions increases at each level of the hierarchy as a function of the order n of the wavelet basis $\mathcal{M}_n$ and the number of push-pull coefficients is the square of the number of basis functions. Working with constant Jacobian mappings thus has a deep impact on performance, since many new hierarchies are created all along the radiosity simulation.

It could be argued that the extra cost ascribable to the use of mappings with non-constant Jacobian must be small for Haar ($\mathcal{M}_1$) wavelets. This is true, since it essentially amounts to recomputing the surface area of a patch. But as we shall see in the results section, the trade-off (in terms of computation time, number of final patches and visual accuracy) is clearly in favor of using $\mathcal{M}_2$ or $\mathcal{M}_3$ wavelets for illuminating objects with varying curvatures and normals. For such bases, using energy-preserving mappings is very important.

5.3 Self-shooting surfaces

Since with the virtual mesh method the input geometry is kept intact, a curved input patch may very well shoot energy onto itself. More precisely, every curved primitive that has at least a concave "island" will self-shoot. For every such surface, we create a *decoy*, i.e., a new surface with the same geometry but zero energy levels.

Handling self-shooting surfaces is then a two-step process. First, the surface shoots energy on its decoy as if it was a different surface. Then the energy levels, residuals and mesh subdivisions of the decoy are copied back to the original surface.

5.4 Managing the hierarchy

Suppose the image of a curved patch by a constant Jacobian mapping is the support of a wavelet approximation of the radiosity function. To benefit from the simple transfer coefficient computation of its virtual support, we proceed as if the virtual support was the real support of the function. All calculations are done on a unit square and so the hierarchical meshing is very natural (see Figure 9). Indeed, a simple quadtree algorithm will produce four identical squares (representing four regions of identical area on $\bar{S}$) at each level which are assumed to be the new virtual supports of the parts of the original shape that fall over them. During subdivision, two special kinds of meshes may appear: empty meshes, which will simply not be considered for emission in future steps, and plain meshes, which can be managed with a classical hierarchical subdivision.

Subdividing the virtual support of a shape rather than the shape itself avoids

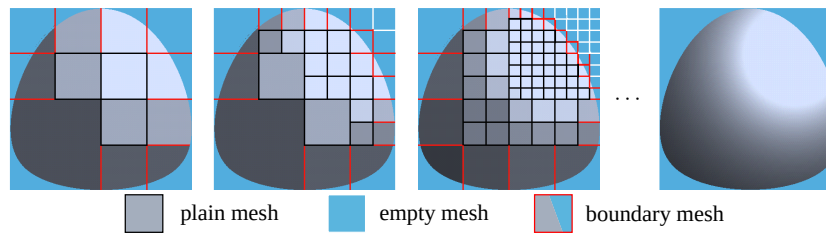

Fig. 9. Hierarchical subdivision of the virtual support of the patch delimited by the curve drawn on the torus of Figure 4.a.

the use of complicated meshing schemes that would produce polygons which would be hard to manage. And the use of the virtual mesh does not simply reduce the number of initial primitives, it also reduces the number of final patches, as will appear clearly in the next section.

6. RESULTS

The wavelet radiosity algorithm with the virtual mesh presented above has been implemented. We now present results obtained with our implementation for various test scenes. Our implementation of the virtual mesh can currently handle complex polygons with arbitrary trimming curves, low-degree curved patches and developable surfaces.

We have used several test cases to illustrate different advantages and properties of the virtual mesh[5]:

- the *opera*, to compare the illumination of scenes with complex polygons to pre-meshed scenes;
- the *teapot*, to compare the illumination of scenes with curved patches to tessellated scenes;
- the *chess set*, to compare the illumination of scenes with curved patches with or without constant Jacobian mappings;
- the *Reality Center*™ and the *Soda Hall*, to show that our implementation can also handle real-size scenes with a reasonable time and memory consumption.

All test cases show that the virtual mesh does make a considerable difference performance-wise and memory-wise.

We have decided not to use any form of clustering for our tests. Our motivations for this were twofold. First, although clustering offers an elegant theoretical solution for reducing the complexity of the radiosity algorithm, its practical use may behave unexpectedly and error control is difficult. Clustering is good to have a quick rough simulation of a scene but not to have simulations that are everywhere very accurate, of the kinds needed in lighting design. Second, we wanted to determine what performance gains could be ascribed to the use of the virtual mesh alone and not to other factors. In addition, since the virtual mesh can also be used in conjunction with clustering, we have good reasons to believe that the benefits of using the virtual mesh observed without clustering would be similar with clustering.

[5]Additional examples and images available at `http://www.vsp-tech.com`.

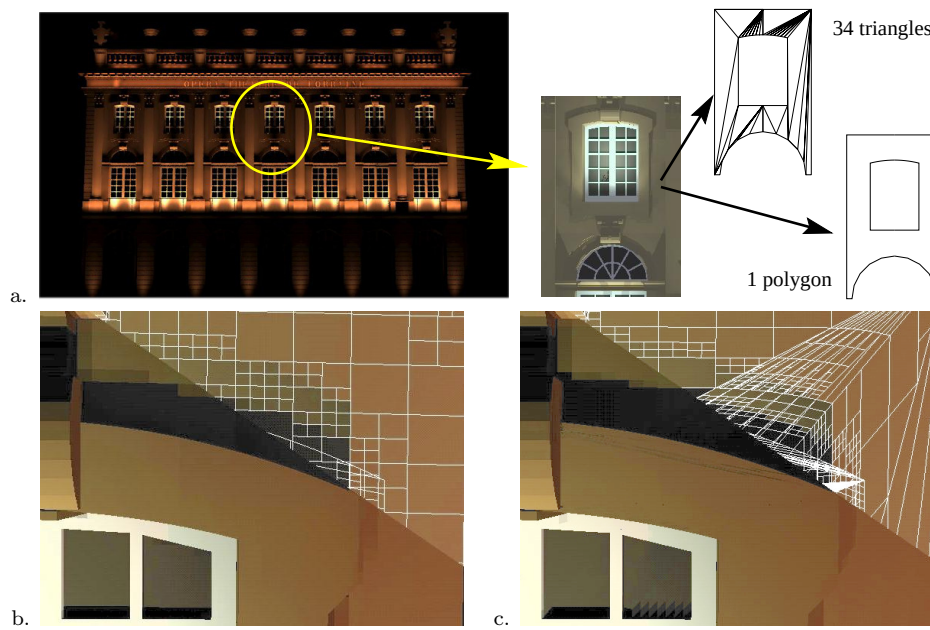

Fig. 10. The opera scene. a. A close-up on the initial mesh of the opera: with simple or complex primitives. b. c. Radiosity solution with (b.) and without (c.) the virtual mesh.

The results were collected on a PC Pentium III 633 MHz with 500 MB of free memory and 500 MB of disk cache. We took care to use similar parameter settings for each test case (in particular, same minimum subdivision size and subdivision oracle threshold). Iterations of the radiosity method are stopped when a user-defined *convergence rate* is reached, defined as the ratio

$$\frac{E_i - E_r}{E_i},$$

where E_i is the initial energy and E_r is the energy remaining to propagate after the current iteration. Finally, the memory results we give are for the difference between the total memory use after simulation minus the resident set size of the process.

6.1 Complex vs. simple polygons

In architectural modeling, many polygons having non-trivial shapes appear. Up to now, these primitives had to be tessellated to be handled by radiosity algorithms. With the virtual mesh, each of these polygons is illuminated as a unique primitive. Our first test case illustrates this property and its impact on visual quality and computation time.

Our first scene, the opera, is a faithful modeling of an opera house of the classical style designed by Emmanuel Héré (18^{th} century). The original model has 17,272 complex polygons. When only simple polygons are allowed, the number of input primitives rises to 32,429 (see Figure. 10.a). Note that the triangulation algorithm used is based on the Open GL tessellator which may generate slivers. Clearly, better polygon triangulators exist (like the constrained Delaunay algorithm). But

geometry	# input prim.	# final meshes	computation time (s)	memory usage (Mb)	# shooting surfaces
simple polygons	32,429	819,109	952	203.5	9,358
complex polygons	17,272	550,159	**482**	**140.3**	6,776
gain	87 %	49 %	98 %	45 %	38 %

Table 1. Results for the opera test scene using Haar wavelets, for a 99 % convergence rate.

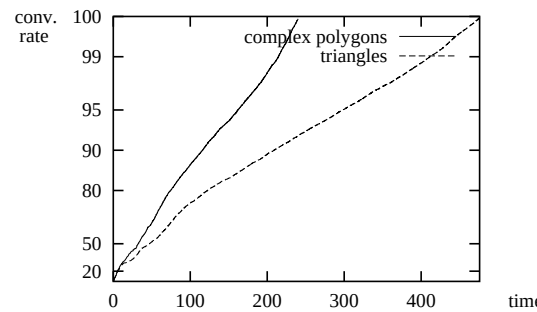

Fig. 11. Evolution of the convergence rate as function of time for the opera test scene.

they generally produce a much larger number of triangles than Open GL. Since the performance of any radiosity algorithm depends heavily on the number of input surfaces, we have decided to stick with the Open GL tessellator so as to have meaningful comparisons.

Qualitatively speaking, Figure 10 shows that the results are visually more pleasing when the virtual mesh is used to handle complex polygons. Clearly, separating the representation of the radiosity function from the geometry and using simple and computationally convenient domains to build the hierarchy yields a much better approximation of radiosity. When the tessellated model is used for the simulation, false geometric discontinuities due to the triangulation may appear, as is the case in Figure 10.c 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: slivers are missed by the visibility algorithm.

Quantitatively speaking, Table 1 shows that the virtual mesh, in addition to improve the visual quality and accuracy of the simulation, also has a strong impact on performance. Indeed, for the same convergence rate of 99 %, the overall computation time is divided by 2. And as shown by Figure 11, the closer to the "ideal" solution one wants to be, the better it gets. In addition, the number of surfaces that have shot energy during the simulation is lower by more than 25 %: this is an important point which saves visibility computation time.

Note that the gain in terms of memory usage (45 %) is less important than what it is for running time. This is easily explained by the fact that subdivisions are not simply due to the shape of the input primitives but also to the complexity of the radiosity function to represent. Here, many subdivisions are due to the complex patterns of shades and shadows around the many windows of the model, thus generating many final meshes and increasing memory usage.

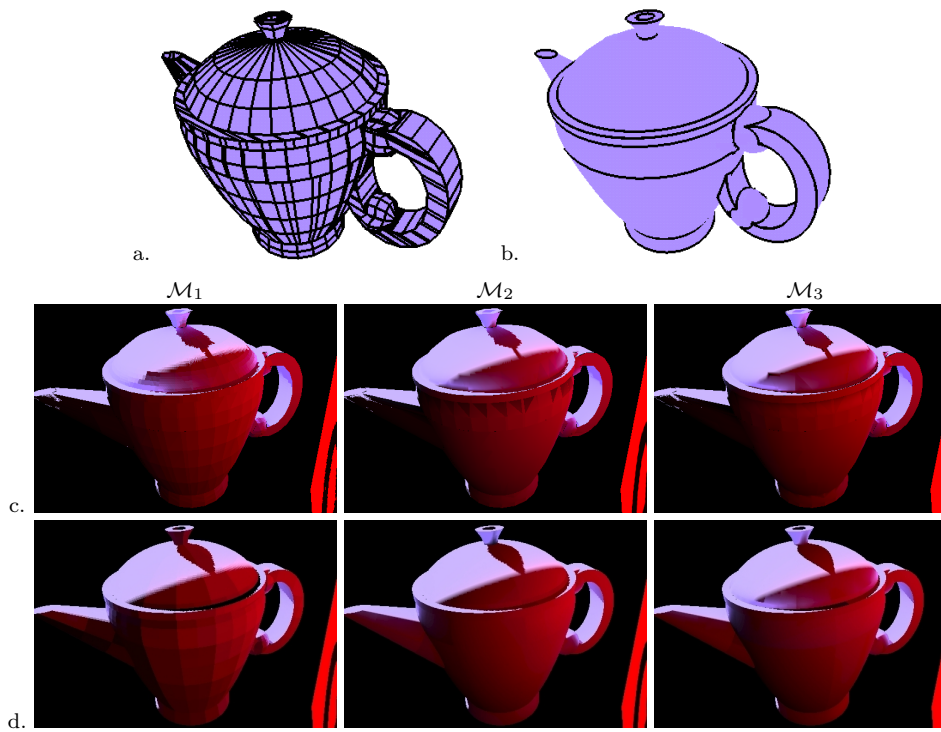

Fig. 12. The teapot test scene. a. The tessellated model. b. The teapot modeled with quadrics. c. Illumination of a. with different wavelet bases. d. Illumination of b. with different wavelet bases.

6.2 Curved vs. tessellated primitives

With the teapot test scene (Figure 12), we illustrate the fact that it is faster and visually more realistic to illuminate scenes made of parametric surface patches directly using the virtual mesh rather than via triangulation.

The teapot is made of 12 quadric surface patches (ellipsoids, cylinders, cones and hyperboloids of one sheet) and 7 complex polygons. Note that some of the patches, notably around the handle, have complex trimming curves. The teapot is illuminated by a white point light source which reflects on a planar red plate, so that there is a lot of indirect lighting. Comparison is made with a tessellation of the teapot. We have chosen a reasonable mesh size (1,392 triangles and parallelograms – see Figure 12.a) so that the teapot looks good enough when viewed from some distance: it is a good compromise between a very coarse mesh (a few hundred primitives) and a mesh fine enough (of the order of 10,000 triangles) to grab the full curvedness of the teapot. Each vertex of the tessellation is equipped with the real normal of the teapot at that point, so that normals can be interpolated during the simulation.

The results of our simulations, all made for the same convergence rate of 99 %, are given in Table 2. For the same wavelet basis ($\mathcal{M}_3$), the illumination of the non-tessellated version of the teapot is more than twice faster than that of the

geometry	wavelet basis	# input primitives	# final meshes	running time (s)	memory usage (Mb)	# shooting surfaces
simple polygons with interpolation of normals	$\mathcal{M}_1$	1,392	52,346	795	28.1	1,300
	$\mathcal{M}_2$		25,475	558	23.6	1,378
	$\mathcal{M}_3$		18,217	605	22.4	1,410
quadrics and constant Jacobian mappings	$\mathcal{M}_1$	19	39,895	583	9.2	21
	$\mathcal{M}_2$		20,920	341	7.8	
	$\mathcal{M}_3$		16,537	**262**	**8.1**	

Table 2. Results for the teapot scene with various wavelet bases, with or without tessellation of the input geometry, for the same global error and the same convergence rate (99 %).

tessellated one. And as illustrated by Figs. 12.c and .d, the visual result is more pleasing when the input surfaces are quadric patches. To obtain a similar visual result, the tessellated teapot would need a very large number of input patches (and thus of final patches, inducing a greater memory usage).

Table 2 shows that the number of final patches is not very different between the two simulations. This is easily explained by the fact that most subdivisions occur along the sharp shadow boundary that appears on the lid of the teapot and that many subdivisions are needed to "grab" the curve, both for the tessellated and the quadrics teapot. However, memory consumption is significantly lower for the illumination with curved patches and the virtual mesh: memory usage is linked to the number of hierarchies created during the simulation and is thus influenced also by the number of input primitives.

Note that the artifacts that one can observe on the tessellated teapot simulation (Fig. 12.c) are due to an acceleration of the algorithm: before considering a patch for subdivision, the oracle tests whether its orientation prevents it from being seen by the emitter by looking at the interpolated normal at a number of Gauss quadrature points. The one artifact that appears near the shadow boundary on the lid is thus the result of a small patch being marked as entirely not visible from the light source, while in fact its top left corner is visible from the source.

The teapot test case also gives another confirmation of the findings of Cuny et al. [Cuny et al. 2000]: the benefits of higher-order wavelets outweigh their costs.

6.3 Constant vs. non-constant Jacobian mappings

The chess set test scene is used to illustrate the impact of the constant Jacobian property on the performance of the wavelet radiosity/virtual mesh algorithm. The scene is made of a chessboard and its chess pieces illuminated by a desk lamp. The chess set alone is made of 555 surface patches, 264 of which are curved (some of them having non-trivial trimming curves, as the knight of Fig. 13). Figure 13 shows a radiosity illumination of the scene with a final pass of ray tracing (3 bounces) that has a small impact on the overall performance (8 seconds for a 256×256 image).

We have compared the performance of two versions of the virtual mesh: using constant Jacobian mappings and using natural coordinates mapping of the input primitives. The main difference lies in the necessity, in the latter case, to compute the push-pull coefficients at each level of the hierarchy, as explained in Section 5.2. Our results are summarized in Table 3. They show that the constant Jacobian requirement is very important. We have determined that the algorithm does not

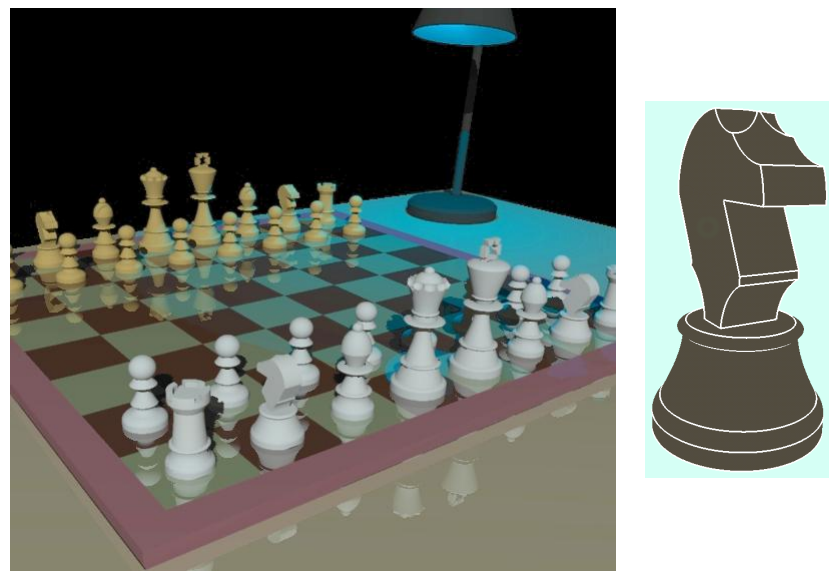

Fig. 13. Illumination of the chess set with constant Jacobian mappings.

mappings	# quadrature points for push-pull	running time (s)	memory usage (Mb)
non-constant Jacobian	5 × 5	304	18.3
	10 × 10	402	
	30 × 30	755	
constant Jacobian	30 × 30	**181**	**8.1**

Table 3. Results for the chess set test scene using linear wavelets, for a 90 % convergence rate.

even converge below 5×5 Gauss quadrature points per support for coefficient estimation and at least 10×10 points are needed to have a satisfactory visual result. Note that it could have been worse for other types of mappings: if a large region in parameter space represents only a small region in equal-area parameter space, then approximation errors can only be compensated by significantly increasing the number of quadrature points. But this has a deep impact on performance, as Table 3 indicates: when a 30×30 "grid" is used, which is the one we take for the computation of the top-level coefficients in the constant Jacobian version, the running time is almost four times slower for the natural coordinates simulation and memory usage is significantly larger.

6.4 Environments with real-world complexity

In this final section, we show that the virtual mesh can also efficiently and robustly handle scenes of high geometric complexity.

The Reality Center™ test scene is a faithful modeling of the SGI immersive visualization facility located in Cortaillod, Switzerland. It is made of 2,561 surfaces, more than half of which are curved (mainly spherical, conical, cylindrical and toroïdal patches), illuminated by 41 point light sources. Figure 14 shows that this

scene	geometry	# input surfaces	# final meshes	running time (s)	memory usage (Mb)
Reality Center™ on PC Pentium III	low-degree patches	2,561	119,486	1,510	157.1
Soda Hall on SGI Origin 2000	simple polygons	272,450	1,124,585	44,585	3,425
	complex polygons	199,550	927,428	30,232	2,612
	quadrics and complex polygons	70,645	827,62	13,411	1,210

Table 4. Results for the Reality Center™ (linear wavelets) and Soda Hall (quadratic wavelets) test scenes, for a 95 % convergence rate.

relatively low number of primitives is enough to represent a complex environment. And we can zoom in on any part of the simulation while keeping a good visual quality. To reach the same visual result with simple polygons, the scene would need to be tessellated in more than 100,000 triangles, substantially hampering the performance. Table 4 summarizes the numerical results of this simulation on our PC Pentium III.

For larger scenes, the PC may not have enough memory for very accurate simulations. To demonstrate that the virtual mesh has no problem with such scenes, we have developed a parallel version of our wavelet radiosity algorithm on a distributed shared-memory machine, the SGI Origin 2000 with 4 195 MHz R10000 MIPS processors and 4 Gb of main memory (the details of the algorithm can be found in [Paul et al. 2000]). Our test scene is the 4^{th} floor[6] of the famous Soda Hall (the computer science building on the campus of the University of California at Berkeley) with its furniture – see Figure 15 for an overview of the illuminated floor. While the original model is made of 272,450 simple polygons, allowing simple curved patches and complex polygons reduces this number to 70,645. Table 4 shows that this falloff does make a difference performance- and memory-wise when the virtual mesh is used, as already observed. The fact that the memory consumption is quite high is a reflection of the fact that there is a lot of data replication in the parallel algorithm.

Note that the images of Figure 1 were made from a simulation of the Soda Hall scene augmented with a special room (room 420A) furnished with several sorts of curved objects (teapot, chess set, benches, jars, desk lamps...).

7. CONCLUSION

In this paper, we have presented a general method for separating the structure of the mesh from the complexity of the geometry in radiosity algorithms. At its core is the virtual mesh, an abstraction of the input geometry that is both simple and computationally convenient. The virtual mesh completely eliminates the effect of often highly tessellated but otherwise simple surface shapes on the algorithm. The method works for both trimmed parametric surfaces and complex polygons. We have shown how to interface the virtual mesh with a wavelet radiosity algorithm. Our implementation provides a hitherto missing link between hierarchical Galerkin radiosity and general trimmed curved surfaces. The virtual mesh has been shown

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

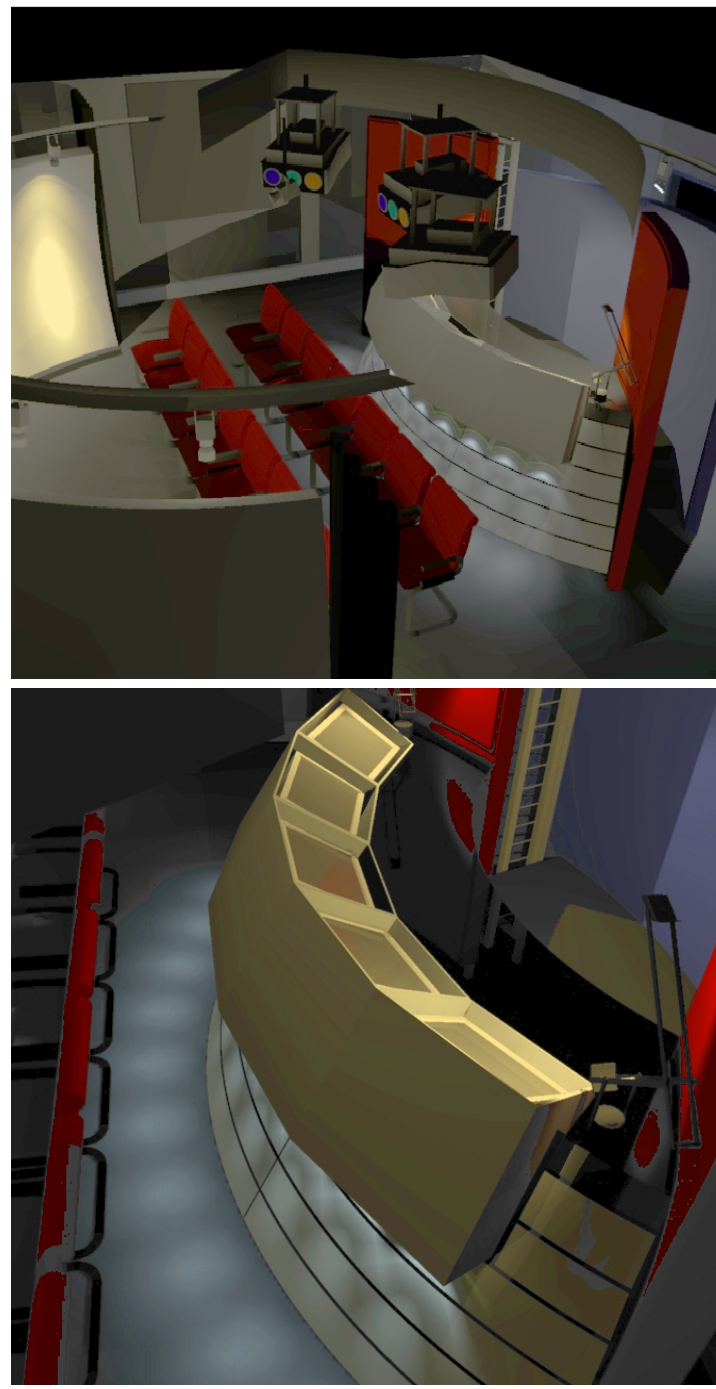

Fig. 14. Simulation of the Reality Center™ test scene.

Fig. 15. Aerial view of the illumination of a floor of the Soda Hall.

to yield very good performances in terms of running time, memory usage and visual quality.

Our method is a new step towards making radiosity robust enough to be employed on useful scenes. Still, several problems remain to be addressed:

- The virtual mesh is not limited to Galerkin radiosity. Indeed, it is a very general method and can be combined with any improved radiosity algorithm. In the future, it will probably see applications to clustering algorithms and to handle glossy surfaces. We are currently working on coupling the virtual mesh with discontinuity meshing, to benefit from the best of the two methods and go one step further in improving the accuracy and realism of the simulations.
- Some triangle meshes (like a laser scan of the Venus de Milo) do not represent simple surface shapes: the underlying surfaces are curved but without a compact mathematical representation. For such surfaces, our method cannot be applied in its current form. However, we are currently investigating ways of globally parameterizing sets of triangles, as discussed in [Lévy and Mallet 1998]. The two main issues here are how to compute energy-preserving mappings for a given set of triangles and how to split a triangle mesh into useful sets of triangles.
- Up to recently, the future of parametric surfaces in computer graphics was open to question. Smoothed versions of polygonal surfaces seemed to take hold. But recently, parametric surfaces received a renewed interest since it was shown that subdivision surfaces are piecewise-parametric (Stam proved this for the Loop and Catmull-Clark subdivision schemes [Stam 1998]). We now need to evaluate how practical our method is for such general surfaces as NURBS and subdivision surfaces. One short-term work is to develop equal-area parameterizations for

cubic Bézier patches.

ACKNOWLEDGMENTS

The authors wish to thank the anonymous reviewers for useful comments and suggestions.

APPENDIX

A. EQUAL-AREA MAPPINGS FOR CURVED PATCHES

Following the ideas developed in Sections 3 and 4, the virtual mesh can be applied to a variety of complex shapes. The only thing that needs to be done to add a new shape to the dictionary of primitives that our wavelet radiosity algorithm can illuminate is to determine an energy-preserving mapping for this shape. We illustrate here several simple but important cases: natural quadrics and flat surfaces.

A.1 Natural quadrics

Because of their fairly important modeling power and modest complexity, quadrics (i.e., algebraic surfaces of degree 2) are important in geometric modeling and visualization. Some quadrics are more important than others. Patches of natural quadrics (planes, cones, spheres and cylinders) make up, according to [Nourse et al. 1980], 85 % of all mechanical pieces.

This section shows how to compute energy-preserving mappings for natural quadrics. Constant Jacobian mappings for the remaining types of quadrics can be found in [Cuny 2000].

Cylinder. Consider a right circular cylinder given in canonical form by:

$$x^2 + y^2 = 1, \quad \begin{cases} x = \cos\theta, \\ y = \sin\theta, \\ z = \varphi. \end{cases}$$

Using the same notations as in Section 3.1, where $\theta = f(u)$ and $\varphi = g(u, v)$, Equation (2) becomes

$$v = c|f'(u)|g(u, v).$$

Thus we can take $\mathcal{J}_\Psi = 1/c = 1, g(u, v) = \varphi = v$ and $f(u) = \theta = u$. In other words, the "natural coordinates" mapping $(x, y, z) \longmapsto (\theta, \varphi)$ is in this case energy-preserving.

Cone. A circular cone can be described by the equation

$$\frac{x^2}{a^2} + \frac{y^2}{a^2} - \frac{z^2}{b^2} = 0, \quad \begin{cases} x = a\cos\theta\varphi, \\ y = a\sin\theta\varphi, \\ z = b\varphi. \end{cases}$$

The constant Jacobian constraint of the mapping $(x, y, z) \mapsto (u, v)$ boils down to

$$v = c\frac{a}{2}\sqrt{a^2 + b^2}|f'(u)|g(u, v)^2.$$

So one can take $\mathcal{J}_\Psi = \frac{a}{2}\sqrt{a^2+b^2}$, $f(u) = u$ and $g(u,v) = \sqrt{v}$ as solution. The mapping

$$(x,y,z) \longmapsto (u,v) = (\theta, \varphi^2)$$

is thus energy-preserving.

Sphere and spheroid. A prolate spheroid is defined by

$$\frac{x^2}{a^2} + \frac{y^2}{a^2} + \frac{z^2}{b^2} = 1, \quad \begin{cases} x = a\sin\theta\sin\varphi, \\ y = a\cos\theta\sin\varphi, \\ z = b\cos\varphi, \end{cases}$$

with $b > a$. Equation (2) writes down as

$$v = c\frac{2}{ab}|f'(u)|\Big(\sin g\sqrt{1-e^2\sin^2 g} + \frac{1}{e}\arcsin(e\sin g)\Big),$$

where

$$e = \sqrt{1-\frac{a^2}{b^2}} > 0$$

is the *ellipticity* of the spheroid. To determine an energy-preserving mapping for this surface, one can take $\mathcal{J}_\Psi = \frac{ab}{2}$ and $u = \theta$. The mapping

$$(x,y,z) \longmapsto \Big(\theta, \sin\varphi\sqrt{1-e^2\sin^2\varphi} + \frac{1}{e}\arcsin(e\sin\varphi)\Big)$$

is thus energy-preserving for the spheroid.

We obtain a mapping for the sphere ($a = b$) as a limit case when $e \to 0$:

$$(x,y,z) \longmapsto (\theta, \sin\varphi).$$

A.2 Flat surfaces

A flat surface is a surface for which the Gaussian curvature vanishes everywhere. Generalized cylinders, generalized cones and tangent developables are popular examples of flat surfaces.

A generalized cylinder is defined by the following parametric equation:

$$\mathbf{x}(\theta,\varphi) = \mathbf{p}\varphi + \mathbf{y}(\theta),$$

where $\mathbf{p}$ is an arbitrary point and $\mathbf{y}(\theta)$ is the parametric equation of the base curve C of the cylinder. The energy-preserving constraint of Equation (2) writes down as

$$v = c|f'(u)|\,\|\mathbf{y}'(\theta)\times\mathbf{p}\|\,g(u,v).$$

If C is planar, such that its equation is $\mathbf{y}(\theta) = (y_1(\theta), y_2(\theta), 0)^T$, and $\mathbf{p} = (0,0,1)^T$ (right generalized cylinder), then this constraint becomes:

$$v = c|f'(u)|\sqrt{y_1'^2(\theta) + y_2'^2(\theta)}g(u,v).$$

For instance, if C is a cardioid, parameterized by

$$\mathbf{y}(\theta) = (a\cos\theta(1+\cos\theta), a\sin\theta(1+\cos\theta), 0)^T,$$

then we can take $\mathcal{J}_\Psi = 2\sqrt{2}a$ and the mapping

$$(x,y,z) \longmapsto (u,v) = (\sqrt{1-\cos\theta}, \varphi)$$

is energy-preserving.

A tangent developable is parameterized by

$$\mathbf{x}(\theta, \varphi) = \mathbf{y}(\theta) + \varphi \mathbf{y}'(\theta),$$

with $\mathbf{y}(\theta)$ is the equation of a base curve C as above. Equation (2) is here

$$v = \frac{c}{2} |f'(u)| \, \|\mathbf{y}'(\theta) \times \mathbf{y}''(\theta)\| \, g(u, v)^2.$$

If C is again a cardioid, then $\Psi^{-1} : (x, y, z) \longmapsto (\theta + \sin\theta, \varphi^2)$ is an energy-preserving mapping such that $\mathcal{J}_\Psi = \frac{3a^2}{2}$.

REFERENCES

Arquès, D. and Michelin, S. 1995. A new radiosity approach for regular objects: application to ruled surfaces. *Computer Graphics Forum 14*, 3, 299–310. Proceedings of Eurographics'95.

Arvo, J., Torrance, K., and Smits, B. 1994. A framework for the analysis of error in global illumination algorithms. *Computer Graphics Proceedings, Annual Conference Series 28*, 4, 75–84. Proceedings of Siggraph'94.

Bekaert, P. and Willems, Y. 1996. Error control for radiosity. In *Proc. of Eurographics Workshop on Rendering, Porto* (1996), pp. 153–164.

Boissonnat, J.-D. and Yvinec, M. 1998. *Algorithmic Geometry*. Cambridge University Press.

Bouatouch, K. and Pattanaik, S. 1995. Discontinuity meshing and hierarchical multiwavelet radiosity. In *Proc. of Graphics Interface'95, Québec, Canada* (1995), pp. 109–115.

Christensen, P., Lischinski, D., Stollnitz, E., and Salesin, D. 1997. Clustering for glossy global illumination. *ACM Transactions on Graphics 16*, 1, 3–33.

Christensen, P., Stollnitz, E., Salesin, D., and DeRose, T. 1996. Global illumination of glossy environments using wavelets and importance. *ACM Transactions on Graphics 15*, 1, 37–71.

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

Cuny, F. 2000. *Radiosité à base d'ondelettes sur des surfaces paramétriques*. Ph. D. thesis, Institut national polytechnique de Lorraine.

Cuny, F., Alonso, L., and Holzschuch, N. 2000. A novel approach makes higher order wavelets really efficient for radiosity. *Computer Graphics Forum 19*, 3, 99–108. Proc. of Eurographics'2000.

Goral, C., Torrance, K., Greenberg, D., and Battaile, B. 1984. Modelling the interaction of light between diffuse surfaces. *Computer Graphics Proceedings, Annual Conference Series 18*, 3, 212–222. Proceedings of Siggraph'84.

Gortler, S., Schröder, P., Cohen, M., and Hanrahan, P. 1993. Wavelet radiosity. *Computer Graphics Proceedings, Annual Conference Series 27*, 221–230. Proceedings of Siggraph'93.

Hanrahan, P., Salzman, D., and Aupperle, L. 1991. A rapid hierarchical radiosity algorithm. *Computer Graphics Proceedings, Annual Conference Series 25*, 4, 197–206. Proceedings of Siggraph'91.

Hasenfratz, J.-M., Damez, C., Sillion, F., and Drettakis, G. 1999. A practical analysis of clustering strategies for hierarchical radiosity. *Computer Graphics Forum 18*, 3, 221–232. Proceedings of Eurographics'99.

Heckbert, P. 1992. Discontinuity meshing for radiosity. In *Proc. of Eurographics Workshop on Rendering, Bristol* (1992), pp. 203–226.

Heckbert, P. and Winget, J. 1991. Finite element methods for global illumination. Technical Report CSD-91-643, University of California, Berkeley.

Holzschuch, N., Cuny, F., and Alonso, L. 2000. Wavelet radiosity on arbitrary planar surfaces. In *Proc. of 11th Eurographics Workshop on Rendering, Brno* (2000), pp. 161–172.

Kaplan, W. 1984. *Advanced Calculus*. Addison-Wesley.

Lévy, B. and Mallet, J.-L. 1998. Non-distorted texture mapping for sheared triangulated meshes. *Computer Graphics Proceedings, Annual Conference Series 32*, 343–352. Proceedings of Siggraph'98.

Nourse, B., Hakala, D., Hillyard, R., and Malraison, P. 1980. Natural quadrics in mechanical design. *Autofact West 1*, 363–378.

Paul, J.-C., Cavin, X., and Alonso, L. 2000. Partitioning and scheduling large radiosity computations in parallel. *Journal on Parallel and Distributed Computer Practices 3*, 3 (September). Special Issue on Parallel and Distributed Computer Graphics.

Requicha, A. and Voelcker, H. 1982. Solid modeling: a historical summary and contemporary assessment. *IEEE Computer Graphics and Applications 2*, 1, 9–24.

Rushmeier, H., Patterson, C., and Veerasamy, A. 1993. Geometric simplification for indirect illumination calculations. In *Proceedings of Graphics Interface'93* (San Francisco, CA, 1993), pp. 227–236. Morgan Kaufmann.

Schäfer, S. 1997. Hierarchical radiosity on curved surfaces. In J. Dorsey and P. Slusallek Eds., *Proceedings of the Eighth Eurographics Workshop on Rendering* (1997), pp. 187–192. Springer Wien.

Schröder, P. 1994. *Wavelet algorithms for illumination computations*. Ph. D. thesis, Department of Computer Science, Princeton University.

Schwarz, C., Teich, J., Vainshtein, A., Welzl, E., and Evans, B. 1995. Minimal enclosing parallelogram with application. In *Proc. 11th Annu. ACM Sympos. Comput. Geom., Vancouver, Canada* (1995), pp. 34–35.

Sillion, F. 1995. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics 1*, 3, 240–254.

Sillion, F. and Puech, C. 1994. *Radiosity and Global Illumination*. Morgan Kaufmann.

Smits, B., Arvo, J., and Greenberg, D. 1994. A clustering algorithm for radiosity in complex environments. *Computer Graphics Proceedings, Annual Conference Series 28*, 435–442. Proceedings of Siggraph'94.

Stam, J. 1998. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. *Computer Graphics Proceedings, Annual Conference Series 32*, 395–404. Proceedings of Siggraph'98.

Stamminger, M., Schirmacher, H., Slusallek, P., and Seidel, H.-P. 1998. Getting rid of links in hierarchical radiosity. *Computer Graphics Forum 17*, 3, 165–174. Proceedings of Eurographics'98.

Stamminger, M., Slusallek, P., and Seidel, H.-P. 1997. Bounded radiosity – Illumination on general surfaces and clusters. *Computer Graphics Forum 16*, 3, 309–317. Proceedings of Eurographics'97.

Willmott, A. and Heckbert, P. 1997. An empirical comparison of progressive and wavelet radiosity. In J. Dorsey and P. Slusallek Eds., *Proceedings of the Eighth Eurographics Workshop on Rendering* (New York, 1997), pp. 175–186. Springer.

Willmott, A., Heckbert, P., and Garland, M. 1999. Face cluster radiosity. In *Proc. of Tenth Eurographics Workshop on Rendering, Granada, Spain* (1999), pp. 293–304.

Winkler, C. 1998. *Expérimentation d'algorithmes de calcul de radiosité à base d'ondelettes*. Ph. D. thesis, Institut national polytechnique de Lorraine.

Zatz, H. 1993. Galerkin radiosity: a higher-order solution method for global illumination. *Computer Graphics Proceedings, Annual Conference Series 27*, 213–220. Proceedings of Siggraph'93.