A FAST INTERPOLATION METHOD TO REPRESENT THE BRDF IN GLOBAL ILLUMINATION

Pierre Y. Chatelier
LAIC, Université d'Auvergne, Aubière, France

Rémy Malgouyres
LAIC, Université d'Auvergne, Aubière, France

Keywords: Global illumination, BRDF, discrete representation, spherical harmonics, control-points, interpolation, rotation.

Abstract: Global illumination that simulates realistic lighting environments makes use of bidirectional reflectance distribution functions (BRDF). Such a function is a surface property, describing how light is re-emitted after hitting this surface. The present paper details a representation of BRDF and radiance distributions, based on control points, that aims at performance. It uses spherical triangles for interpolation, least square mean or linear programming for optimal representation, and very fast rotation by precomputing control points reweightening. It has some more advantages for deforming an existing shape. This approach is compared to Spherical Harmonics, and outperforms this last one.

1 INTRODUCTION

Global illumination that simulates realistic lighting environments must be able to describe how light is re-emitted after hitting the surface of an object, depending on the optical properties of this surface: the *material* properties. The Lambertian model is one of the simplest: a fraction of light (depending on the zenithal incidence) is isotropically re-emitted above the surface. This is also known as the *ideal diffuse case*. In this case, the material property is embedded in a single factor, as a real number.

However, most objects, like mirrors, velvet, wet surfaces... should not use such a simple model. First, asymetric materials like velvet must take in account not only the zenithal angle of an incoming ray, but also the azimuthal orientation of the ray according to the surface. Second, incoming light is anisotropically re-emitted. This is particularly obvious for a perfect mirror, which is said to be a perfect *specular* reflector. Implementing a material mixing up diffuse and specular properties implies a way to represent spatial anisotropy.

The notion of Bidirectional Reflectance Distribution Function (BRDF), and radiance distribution, is the common way to describe such a behaviour. This paper describes the well known problem of representing and using BRDFs (Neumann, 2001; Rusinkiewicz, 1997; Rusinkiewicz, 1998). The required operations for using them in a global illumination solver are presented, and we show how they are addressed with spherical harmonics. Then, we introduce a fast interpolation method, designed to perform these operations with maximal performance and a reasonable precision. Computational experiments have been performed to compare this method with the spherical harmonics.

2 USING A BRDF

2.1 Representation

A BRDF function is a function of two parameters: the direction of incoming light, and the direction of the outgoing ray for which we want to know the re-emission. Each direction is in fact a two-dimensional parameter, usually denoted by the two spherical angles θ and ϕ, representing elevation and azimuth (see Fig. 1), so that a BRDF is rather seen as a 4-dimensional function . The returned value is the ratio, of the reflected (or transmitted) radiance in the given outgoing direction, to the incoming energy flux from the incoming direction. The BRDF of an ideal-diffuse surface is constant, while an ideal reflector can be represented by a Dirac delta function. A simple BRDF may be stored as a mathematical function, but arbitrary BRDFs usually require a set of sample points that have to be interpolated. Many representations for BRDFs have been proposed in the computer graphics literature (Ward, 1992; Ashikhmin and Shirley, 2000; Kautz et al., 2002), generally based on a decomposition on a basis of spherical functions, like spherical harmonics or spherical wavelets (Schröder and Sweldens, 1995; Koenderink et al., 1996).

2.2 Models and Instanciations: Brdf and Radiance Distributions

By stating that we are using a BRDF to modelize light re-emission, we are only describing how we handle this material property. However, one must not forget that this material property is static and is different from the *effective emission* that appears in each element that constitutes the surface. Indeed, considering the 3D scene for which we are computing global illumination, it is usually discretized into several patches, each patch behaving like a light emitter. Patches from the same surface will share the same BRDF, that *describes* their behaviour. But the *effective* emission varies from patch to patch, according to the quantity of energy that each one is re-emitting, and that has been computed by the global illumination algorithm. This effective emission is the *radiance distribution*, which is a function of one parameter: the outgoing direction (see Fig. 2). Here again, it can be seen as a two-dimensional spherical function on θ and ϕ. This function is not known in advance, it is what is computed by the global illumination algorithm. At first, only the light sources have non-zero radiance distributions. Then, progressively, light propagates and hits some patches, that affects their radiance distributions according to their local BRDF. Solving the global illumination problem means that BRDF are known 4-dimensional functions, while radiance distributions are the unknowns, 2-dimensional functions.

For the sake of performance, these two kind of objects are usually represented with the same data structures. Instead of encoding a 4-dimensional function, the BRDF can be implemented as the interpolation of 2-dimensional functions. That is to say, $f(\theta_{in}, \phi_{in}, \theta_{out}, \phi_{out})$ is computed from some $g_{\theta_{in},\phi_{in}}(\theta_{out}, \phi_{out})$. Hence, both BRDF and radiance distributions can be represented by 2D spherical functions, and we only need a way to encode efficiently such data.

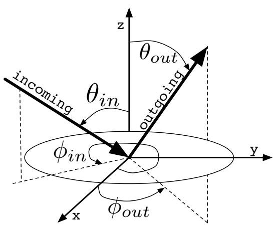


Figure 1: Polar angles to define a BRDF. θ is the zenithal angle (elevation), ϕ is the azimuthal angle (orientation).

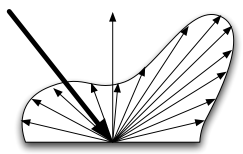


Figure 2: A radiance distribution is the response of a BRDF for some incoming light. This one has diffuse and specular behaviour.

2.3 Operations

As it is stated in Section 2.2, managing radiance distributions requires to encode 2D spherical functions. In order to use such data, we must ensure some operations to be easy to perform on the representation. These operations are querying value, scaling, adding, and rotating the distributions. Since they are to be performed very often, they must be very efficient.

2.3.1 Querying Value

The radiance distribution of a scene element evolves while this element receives light during global-illumination iterative solving. We must be able to query at any time the light leaving an element in a particular direction. This can be done using a mathematical model, which returns a value for some parameters. This can also be done with interpolation, which tries to guess a value according to some neighborhood. The Spherical Harmonics mathematical model is described in Section 2.4. We propose in Section 3 of this paper an interpolating method.

2.3.2 Scaling and Adding

When some light hits a scene element, its local BRDF is queried for the matching radiance distribution. Then this distribution is scaled by the effective incoming energy quantity, and added to the radiance distribution already held by the element. See Figure 3 for an illustration.

2.3.3 Rotating

Radiance distibutions returned by BRDF queries are computed in a global system of coordinates. They have to be rotated to be added to a scene element, which uses a local system. Indeed, θ is related to the

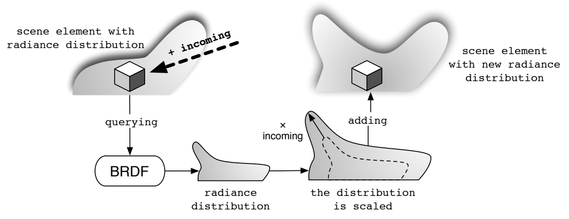

scene element with radiance distribution
+ incoming
scene element with new radiance distribution
querying
adding
× incoming
BRDF
radiance distribution
the distribution is scaled

Figure 3: Scaling and adding radiance distribution.

normal vector of the surface, and the azimuthal angle is meaningful if the surface has an orientation.

2.4 Common Solution

A common method to fulfill all previous requirements is to use a decomposition of the radiance distribution function on a basis of spherical functions. The basis being known, the radiance distribution is just encoded by a set of coefficients. Adding and scaling distributions becomes as easy as adding and scaling the coefficients. Rotation is usually the stumbling block.

2.4.1 Spherical Harmonics

Spherical harmonics are a basis of functions that suits very well at representing spherical 2D functions (François X. Sillion and Claude Puech, 1994; Kautz et al., 2002). The spherical harmonics basis is denoted by $Y_{l,m}$ and a function f is decomposed with $C_{l,m}$ coefficients as

$$f(\theta,\phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} C_{l,m} Y_{l,m}(\theta,\phi) \tag{1}$$

The definition of $Y_{l,m}$ functions makes use of associated Legendre Polynomials $P_{l,m}(x)$ and a normalizing constant $N_{l,m}(x)$.

$$Y_{l,m}(\theta,\phi) = \begin{cases} N_{l,m}P_{l,m}(\cos\theta)\cos(m\phi) & \text{if } m > 0 \\ N_{l,0}P_{l,0}(\cos\theta)\sqrt{2} & \text{if m = 0} \\ N_{l,m}P_{l,-m}(\cos\theta)\sin(-m\phi) & \text{if } m < 0 \end{cases}$$

$$N_{l,m} = \sqrt{\frac{2l+1}{2\pi}\frac{(l-|m|)!}{(l+|m|)!}}$$

$$P_{l,m}(x) = \begin{cases} (1-2m)\sqrt{1-x^2}P_{m-1,m-1}(x) & (l=m) \\ x(2m+1)P_{m,m} & (l=m+1) \\ x\frac{2l-1}{l-m}P_{l-1,m}(x) - \frac{l+m-1}{l-m}P_{l-2,m}(x) & \text{(otherw.)} \end{cases}$$

2.4.2 Cost of the Representation

The representation with spherical harmonics requires to decide how many coefficients are kept for the decomposition. Since BRDF are usually rather smooth, only a few dozens coefficients will be needed. it is important to keep this number the smallest possible, because since each scene element will hold a whole set of coefficients, memory becomes rapidly a critical resource. A good balance must be found between the expected precision and the required number of coefficients (see Fig. 4). For a real function, only real coefficients are needed.

Ideal specular reflectors are typical examples of failure of the assertion about smooth BRDFs. However, such reflectance should not be handled by a very high frequency distribution, but algorithmically (Sillion et al., 1991).

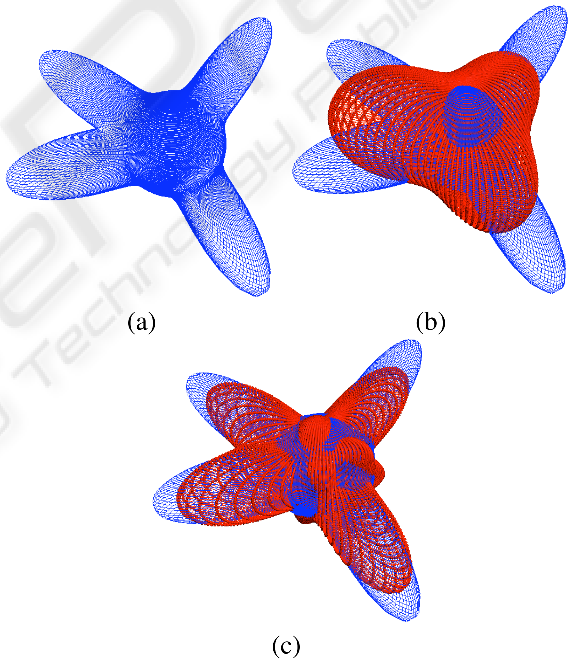


Figure 4: (a) A spherical shape that is to be approximated. It is made of a sphere with some stretchings. (b) Spherical harmonics, 16 coefficients (dotted shape). This is not enough to approximate the shape. (c) Spherical harmonics, 64 real coefficients (dotted shape). This is a good approximation of the spherical shape.

2.4.3 Rotation with Spherical Harmonics

The rotation with spherical harmonics is known to be the stumbling block of this representation (Křivánek et al., 2005). It is proved to be a linear operation (Öhrström, ; Blanco et al., 1997), by multiplying the vector of coefficients by a particular matrix.

However, such a matrix may be unpractical in the circumstances (Erdelyi, 1953).

3 CONTROL-POINTS BASED BRDF

In order to achieve maximal performance, we propose a method based on interpolation on spherical triangles. This method requires a reasonable amount of coefficients, is precise enough for smooth radiance distributions, achieves very good performance for common operations, and suits very well for a direction-based global illumination solver.

3.1 Interpolation

This simplest way to represent a spherical 2D function is to sample it on some points. If the polar angles of these points are fixed, the representation only needs to store the radius component of the spherical coordinates of these points. Then, two difficulties must be adressed: choose the set of control points, and choose an interpolation algorithm.

3.1.1 Choosing the Control Points

The control points must be chosen regularly on the sphere, so that no region is arbitrarily finer than another. Once dispatched on the sphere, the control points are radially translated to represent spherical shapes. A good method to achieve this repartition is to recursively subdivide a regular polyhedron (Schröder and Sweldens, 1995). For instance, starting from a icosahedron (12 vertices), one can take the middle of each edge to divide each face into four new triangles.

The initial polyhedron is an important parameter for the number of control points you wish to have. With an icosahedron, the number of points grows rapidly. At orders 0, 1, 2, 3 of subdivisions, this number is respectively 12, 42, 162, 642 (Fig. 5 and 6). As stated in section 2.4.2, each scene element holds a whole set of coefficients, so that its cardinal should be kept as small as possible, while being big enough to allow precise radiance distributions. Our experiments has shown that 162 points are a good compromise (Fig 7). On these pictures, the dots are not the control points, but some samples interpolated by our method, using the control points.

3.1.2 Weightening

By subdivising as described above, we can define how a point is interpolated by the control points.

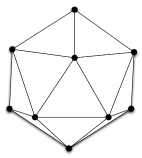


Figure 5: Icosahedron (order 0 of subdivisions).

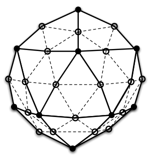


Figure 6: Starting with icosahedron, order 1 of subdivisions.

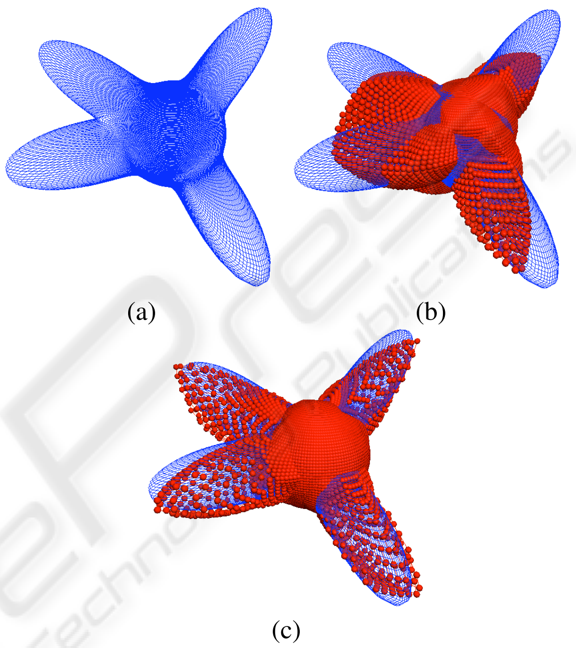


Figure 7: (a) A spherical shape that is to be approximated. It is made of a sphere with some stretchings. (b) 42 control points are used. The result is clearly not precise enough to approximate the given shape. (c) 162 control points are used. This is a good approximation of the spherical shape.

Let P be a point to be interpolated. If we project P, and the control points, on the unit sphere, then the projection P' of P falls into a spherical triangle made by three of the projected control points on the sphere. P' can also lie on an edge or a vertex in degenerated cases, but this is not a problem.
Instead of using classical bi-linear interpolation, that is useful for rasterized triangles in computer graphics, we can instead weighten the vertices by the ratio of the triangles areas defined by P' and the control points of the neighborhood. These triangles are spherical triangles and their area is easy to compute with respect to the formula:

$$area(ABC) = \widehat{A} + \widehat{B} + \widehat{C} - \pi \qquad (2)$$

where $\widehat{A}, \widehat{B}, \widehat{C}$ are the angles at the vertices.
Let P be the point to be interpolated, its projection P'

falling into the spherical triangle defined by the projections C'_1, C'_2, C'_3 of the control points C_1, C_2 and C_3. We denote the interpolation by writing

$$P = \omega_1 C_1 + \omega_2 C_2 + \omega_3 C_3 \qquad (3)$$

the unknowns being the weights ω_i.
These weights can be computed according to the areas of the spherical triangles formed by the projections. Let these triangles be denoted by $P'C'_1C'_2$, $P'C'_1C'_3$, $P'C'_2C'_3$, their areas being respectively $\alpha_1, \alpha_2, \alpha_3$.
We can set

$$\omega_i = \frac{\alpha_i}{\alpha_1 + \alpha_2 + \alpha_3} \qquad (4)$$

to get a normalized weightening system (Figure 8a).

3.1.3 Finding the Triangle

To determine the three control points to be used in interpolation, we have to quickly compute in which triangle a point is falling. If we are given the polar angles of this point, it is easy to determine some possible triangles where it is projected. This only requires the control points to be sorted by parallels and meridians. Once we have found the two parallels encloising the projected point, we can determine which neighbour control points on these parallels surround the projection. Most of the time, more than one triangle are candidate, but some cross products can then determine easily which is the good one (Figure 8b).

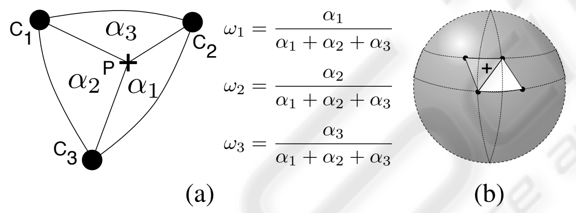


Figure 8: (a) Interpolation on spherical triangles. The weights w_i are determined from the areas α_i. (b) The parallel and meridians of the control points help finding candidates holding the projection.

3.2 Representing an Existing Function

We explained in Section 2.2 that even 4D-BRDFs are modelized using 2D-spherical functions, unifying BRDF and radiance distributions representations. Therefore, we must provide an efficient way to approximate a given 2D function, by finding an optimal set of control points regarding the interpolation algorithm.

To find such an optimal set, we can solve the problem using Least-Squares Method, or Linear Programming. Let f be the 2D spherical function to interpolate, and $P_i(r_i, \theta_i, \phi_i), i \in [0;n]$ the samples of this function. Let $C_j(r_j, \theta_j, \phi_j), i \in [0;m]$ be the control points ($n > m$ is expected). The unknowns are the r_j. Our interpolation method states that P_i is interpolated by at most three control points C_1, C_2 and C_3. We denote that by $r_i = \omega_{i,1} r_1 + \omega_{i,2} r_2 + \omega_{i,3} r_3$.
Let X be the unknown vector $(r_j)_{j \in [0;m]}$ of radial coefficients of the control points C_j. Let Y be the vector $(r_i)_{i \in [0;n]}$ of radial coefficients of the samples P_i. Let A be the matrix of the weights of interpolation. Each line of that matrix is the interpolation of one sample, and holds at most three non-null weights.
The system to solve is $AX = Y$. Usually, this system is over-determined since we get as many samples as we want, while the number of control points remains limited. It has likely no solution and only an approximation is needed.

3.2.1 Least-Squares Method

To solve this system, we can use the Least-Squares Method and set $X = (A^t A)^{-1} Y$. This has shown very good results on smooth functions.

3.2.2 Linear Programming

A common problem of the least-squares method is that there may be a high local error as long as the error is low elsewhere. In (Marzais et al., 2006), this problem is addressed using linear programming. By minimizing the infinite norm of the error, this method tries to output results with a locally limited error.
For our problem, that linear programming method consists in solving Equation 5 to minimize the infinite norm of the error.

$$-\mathbb{1}.h \leq ((AX - Y)_i)_{i \in [0;n]} \leq \mathbb{1}.h \qquad (5)$$

h is the error that is to be minimized

However, better results are obtained in our case by minimizing the infinite norm of the *relative* error (Equation 6). This causes small scale areas to be less perturbed, at the cost of less precise big scale areas.

$$-\mathbb{1}.h \leq \left(\frac{(AX - Y)_i}{Y_i} \right)_{i \in [0;n]} \leq \mathbb{1}.h \qquad (6)$$

As a compromise, we obtained best results by dividing with the square root of the value (Equation 7).

$$-\mathbb{1}.h \leq \left(\frac{(AX - Y)_i}{\sqrt{Y_i}} \right)_{i \in [0;n]} \leq \mathbb{1}.h \qquad (7)$$

Indeed, as it can be seen on Figure 7 (c), some areas are very well approximated, while other are less precise. If the maximal error is found around the peaks, it may be a small error compared to the peak scale, but a big error compared to the small values of the shape.

Thus, if the error cannot be less than ε, this may cause perturbations in the range $[-\varepsilon;\varepsilon]$. This would spoil small scale areas for which ε is not a small value. By taking the relative error, each area is constrained to be as good as possible. This is illustrated in Fig. 9, with a large peak as shape.

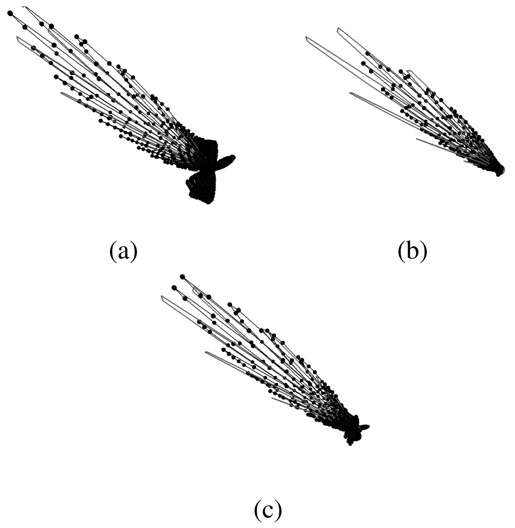


Figure 9: (a) Linear solving with absolute error may cause small scale areas to be perturbed. (b) Linear solving with relative error improves small scale areas but is less precise on big scale ones. (c) A compromise can be found with the relative error by dividing by the square root of the value instead of the value itself.

After many tests, the least-squares method kept performing a little better than the linear programming method. Both methods are easy to implement and may be of interest for complex shapes.

3.3 Deforming Existing Shapes

The set of coefficients characterizing the decomposition of a function, on a basis of functions, cannot be easily interpreted. Most of the time, the first coefficients characterize the shape globally, and additional coefficients introduce local details; but deforming the shape might have an impact on every coefficient. Therefore, it not easy to deform an existing shape, just by acting on the coefficients, since the consequences are difficult to foresee.
Our interpolation method has the particularity that any point is characterized by at most three coefficients that are very easy to determine. Pulling or pushing a point away from the center of the shape has an effect only on these three coefficients. To change the radius of a point of an existing shape, it is only necessary to recompute the weights of the three associated control points. Let us consider $P(r,\theta,\phi)$ interpolated by $C_1(r_1,\theta_1,\phi_1)$, $C_2(r_2,\theta_2,\phi_2)$ and $C_3(r_3,\theta_3,\phi_3)$. This is denoted by $r = \omega_1 r_1 + \omega_2 r_2 + \omega_3 r_3$
Let $P'(r+\delta r,\theta,\phi)$ be the new point P. Then we must find δr_1, δr_2 and δr_3 such that $r+\delta r = \omega_1(r_1+\delta r_1) + \omega_2(r_2+\delta r_2) + \omega_3(r_3+\delta r_3)$.
A trivial soluion is $\delta r = \delta r_1 = \delta r_2 = \delta r_3$, since

$$\omega_1(r_1+\delta r) + \omega_2(r_2+\delta r) + \omega_3(r_3+\delta r) = \underbrace{\omega_1 r_1 + \omega_2 r_2 + \omega_3 r_3}_{r} + \delta r \underbrace{(\omega_1+\omega_2+\omega_3)}_{1}$$

However, it is a bad solution since it is just a shift of the spherical triangle in which P is falling. The risk is to perturbate other interpolated point too much. A better approach is to weighten δr_i by ω_i, so that a control point is shifted according to its importance in the interpolation of P.
For instance, if we require δr_i to be proportional to $\omega_i r_i$, we just have to divide by $\sum \omega_j^2$ to normalise. Hence, the solution we chose is :

$$\delta r_i = \frac{\omega_i}{\sum\limits_j \omega_j^2} \delta r \qquad (8)$$

3.4 Rotation

To perform an efficient interpolation, our method does not allow the angles of the control points to change. Therefore, the rotation cannot be applied to the control points. It is possible, but too expensive, to recompute the control points by resampling the function, rotating the samples, and reapplying Least-Squares Method (or Linear Programming) to get the new control points. The rotation being a critical function, it should be more efficient than that.

3.4.1 Permutation Approach

In our control points system, a rotation can be approximated by a permutation of the control points, by assigning to each *virtually rotated* control point, one (another) of the (non-rotated) control points, most likely the closest (see Figure 10).
Such a process is very fast since the rotation is then reduced to a set of pre-computed permutations for relevant angles. The drawback is that the rotation is no more a continous process, can be transformed into the Identity for small angles, and that much error can be accumulated in a sequence of rotations.

To compute the permutation associated to a rotation, a naive method would be to match each virtually rotated control point to the closest control point, starting from the best matching (with minimal error) and ending with the worst one. The problem is that it can

end with catastrophic matchings since less and less control points remain free while the matchings occur (see Figure 11).
To overcome the problem, another strategy can be adopted: first, the best matching is realized with a control point *C*. Then each matching is done the best possible, starting from the point which is the closest to *C* and ending with the furthest one. That prevents a point to become isolated, with all its neighbours having realized a match (monopolizing the control points around). This would be the cause of a poor matching.

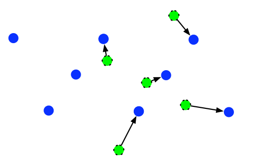


Figure 10: Matching each virtually rotated control point (dotted stroke) to the closest non-rotated point (plain stroke).

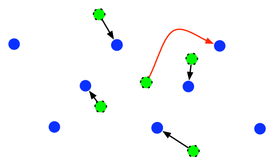


Figure 11: "Best-matchings first" may end in catastrophic matchings if not enough control points remain free. Here, the central point is poorly matched.

3.4.2 A Continuous Approach

The best representation of the rotated shape would be found by resampling it, and perform LSM again to solve the constrained system. It is not affordable, but makes it obvious that the weights of the control points should be reconsidered rather than merely permuted. As a matter of fact, it is rather easy to perform, by *interpolating the virtually rotated control points* on the original, non-rotated, control points.
Each control point, when rotated, can be seen as a sample, that is to be interpolated by three "real" control points. Thus, a "real" control point can be involved in the interpolation of many "virtually rotated" control points. It is given one weight per interpolation. Then, its new radial distance is set according to a weighted sum of the interpolations for which it is participating. Its new position is a compromise and does not necessarly satisfy each interpolation, but this simplified processus gives us a simple, continuous rotation method.

3.5 Direction-Driven Global Illumination Solver

Global illumination algorithms usually use patches as scene elements, and gathers for each patch the quantity of light received by other patches visible from there. By iterating this algorithm, light propagates in the scene and converges towards a radiosity solution.

Another approach presented in (Chatelier and Malgouyres, 2006) uses a special method to compute visibility between volumetric scene elements, and the radiosity algorithm is then parametrized by a set of directions in space to consider. Light is transported only on these directions. This comes from the fact that visibility computations are factorized along some rays (Figure 12).
A consequence of this direction-driven radiosity solver is that it is also possible to factorize computations that are done along a ray. Indeed, the interpolation method described in the present paper computes weights for a given direction (θ, ϕ). Combined with the direction-driven solver, this computation can be factorized for all the scene elements that lie along a ray. It results in an additional optimization that our method brings to global illumination solving.

This is not a marginal optimization. We have also used our BRDF representation in the *radiative transfer* problem (Lenoble, 1993); in this problem, a cloud is illuminated and the energy propagations occur according to some distributions that are called *phase functions*. These phase functions can be represented by 2D spherical functions. Only one global coordinate system is used, since the volumic elements that are representing the cloud are not oriented. So, no rotation is needed. In this context, if *N* voxels are encoding the cloud, each one is queried for its radiance distribution in the current direction (θ, ϕ). Then, the computation of the weights for (θ, ϕ) can be factorized for the *N* voxels.

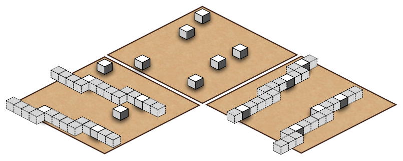


Figure 12: A direction-driven algorithm can factorize the weights computation of all elements lying along a ray, when querying values from radiance distributions.

4 EXPERIMENTAL RESULTS

To compare our implementation of BRDFs using control points, with an implementation using Spherical Harmonics, we have used our radiosity solver (Chatelier and Malgouyres, 2006). The spherical harmonics relies on the SpharmonicKit (Kostele and Rockmore, 2004), which performs very well, so that the performance gap should not be a defect of our implementation of the spherical harmonics.
Figure 13 shows that our method largely outperforms the spherical harmonics approach. Moreover, we did not implement the rotation of the spherical

harmonics, so that it is not even slowed down by their known bottleneck. The scene that has been used is a very simple assemblage of cubes, with diffuse BRDFs. It has been discretized into 13000 voxels for the solver.

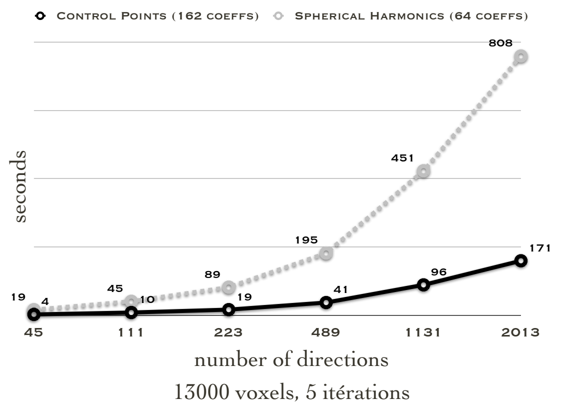
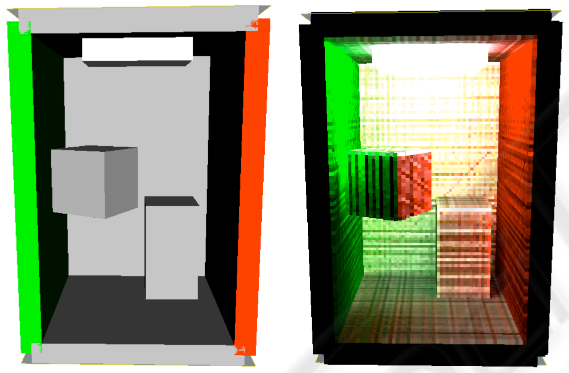


Figure 13: The performance of the control points method (plain stroke) is far better than the spherical harmonics.



(a) (b)

Figure 14: (a) Simple scene. (b) Raw data per voxel after radiosity solving.

REFERENCES

Ashikhmin, M. and Shirley, P. (2000). An anisotropic Phong BRDF model. *Journal of Graphics Tools: JGT*, 5(2):25–32.

Blanco, M. A., Florez, M., and Bermejo, M. (1997). Evaluation of the rotation matrices in the basis of real spherical harmonics. *J. Molecular Structure (Theochem)*, 419:19–27.

Chatelier, P. Y. and Malgouyres, R. (2006). A low-complexity discrete radiosity method. *Computers & Graphics*, 30:37–45.

Erdelyi, A. (1953). *Higher transcendental functions*, volume 2. McGrawHill book company, Inc.

François X. Sillion and Claude Puech (1994). *Radiosity & Global Illumination*. Morgan Kaufmann Publishers, Inc.

Kautz, J., Sloan, P.-P., and Snyder, J. (2002). Fast, arbitrary brdf shading for low-frequency lighting using spherical harmonics. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 291–296.

Koenderink, J. J., van Doorn, A. J., and Stavridi, M. (1996). Bidirectional reflection distribution function expressed in terms of surface scattering modes. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume II*, pages 28–39, London, UK. Springer-Verlag.

Kostele, P. and Rockmore, D. (2004). The SpharmonicKit and S2Kit libraries. http://www.cs.dartmouth.edu/ geelong/sphere.

Křivánek, J., Konttinen, J., Bouatouch, K., Pattanaik, S., and Žára, J. (2005). Fast approximation to spherical harmonic rotation. In *SCCG '06: Proceedings of the 22nd spring conference on Computer graphics (to appear)*, New York, NY, USA. ACM Press.

Lenoble, J. (1993). *Atmospheric Radiative Transfer*. A. Deepak Publishing.

Marzais, T., Grard, Y., and Malgouyres, R. (2006). $lp-$fitting approach for reconstructing parametric surfaces from point clouds. In *International Conference on Computer Graphics Theory and Applications, Setúbal, Portugal*, pages 325–330.

Neumann, A. (2001). *Constructions of Bidirectional Reflection Distribution Functions*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria.

Öhrström, M. Spherical harmonics, precomputed radiance transfer and realtime radiosity in computer games. http://citeseer.ist.psu.edu/685526.html.

Rusinkiewicz, S. (1997). A survey of brdf representation for computer graphics. http://www.cs.princeton.edu/˜smr/cs348c-97/surveypaper.html.

Rusinkiewicz, S. (1998). A new change of variables for efficient BRDF representation. In Drettakis, G. and Max, N., editors, *Rendering Techniques '98 (Proceedings of Eurographics Rendering Workshop '98)*, pages 11–22, New York, NY. Springer Wien.

Schröder, P. and Sweldens, W. (1995). Spherical wavelets: efficiently representing functions on the sphere. *Computer Graphics*, 29(Annual Conference Series):161–172.

Sillion, F. X., Arvo, J. R., Westin, S. H., and Greenberg, D. P. (1991). A global illumination solution for general reflectance distributions. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 187–196, New York, NY, USA. ACM Press.

Ward, G. J. (1992). Measuring and modeling anisotropic reflection. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 265–272, New York, NY, USA. ACM Press.