Fast Approximation to Spherical Harmonic Rotation

Jaroslav Křivánek* Czech Technical University
Jaakko Konttinen† Univ. of Central Florida
Sumanta Pattanaik‡ Univ. of Central Florida
Kadi Bouatouch§ IRISA/INRIA Rennes
Jiří Žára¶ Czech Technical University

Abstract

Rotation of functions represented by spherical harmonics is an important part of many real-time lighting and global illumination algorithms. For some of them a per-vertex or even per-pixel rotation is required, which implies the necessity of an efficient rotation procedure. The speed of any of the existing rotation procedures is, however, not able to meet the requirements of real-time lighting or fast global illumination. We present an efficient approximation of the spherical harmonic rotation applicable for small rotation angles. We replace the general spherical harmonic rotation matrix by its truncated Taylor expansion, which significantly decreases the computation involved in the rotation. Our approximation decreases the asymptotic complexity of the rotation—the higher the order of spherical harmonics, the higher the speed-up. We show applications of the proposed rotation approximation in global illumination and real-time shading. Although the rotation approximation is accurate only for small rotation angles, we show this is not a serious limitation in our applications.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Shading, Shadowing

Keywords: spherical harmonics rotation, global illumination, radiance caching, environment mapping, normal mapping

1 Introduction

When using spherical basis functions (e.g. spherical harmonics or wavelets) for real-time shading with environment lighting , one has to face the problem of aligning the environment lighting (represented in the global coordinate frame) with the reflectance function, or BRDF (represented in the local coordinate frame at each surface point). The alignment is achieved through a rotation of a (hemi)spherical function. Kautz et al. [2002] perform the rotation of the environment lighting represented by spherical harmonics for each vertex during real-time rendering and report that the rotation is the bottleneck. Ng et al. [2004] avoid the rotation by storing the BRDF multiple times, pre-rotated to the global frame for different surface normal directions. This approach is memory demanding, prohibits the use of high frequency BRDFs and does not allow anisotropic BRDFs. Pre-computed radiance transfer (PRT) on rigid objects with smooth surfaces [Sloan et al. 2002; Lehtinen and Kautz 2003; Sloan et al. 2003; Wang et al. 2004; Liu et al. 2004; Wang et al. 2005] avoids the rotation problem since the alignment is included in the transfer matrix stored per-vertex. However, in the case of non-smooth surfaces, where the surface normal is modulated by a texture, a per-pixel rotation is needed even for PRT. Also for global illumination computation, the efficiency of rotation of functions represented by spherical harmonics can be critical. Namely in *radiance caching* [Křivánek et al. 2005; Křivánek 2005], the speed of the spherical harmonics rotation procedure determines the efficiency of illumination interpolation.

To our knowledge, no simple rotation procedure exists for wavelet representation. Functions represented by spherical harmonics can be rotated by a linear transformation of the representation coefficients [Green 2003], but the existing procedures [Ivanic and Ruedenberg 1996; Ivanic and Ruedenberg 1998; Choi et al. 1999; Kautz et al. 2002] are slow and therefore cause a bottleneck in rendering algorithms. Representation by Zonal harmonics [Sloan et al. 2005] or by a sum of Gaussians [Green et al. 2006] can be rotated efficiently, but the slow non-linear optimization needed to fit a given function limits their use only to scenarios where a lengthy precomputation is tolerable.

In this paper we address the rotation of functions represented by spherical harmonics. We propose an efficient approximation of the spherical harmonic rotation based on replacing the general spherical harmonic rotation matrix with its truncated Taylor expansion. The proposed rotation approximation is faster and has a lower computational complexity in terms of spherical harmonic order than previous methods. Unfortunately, our approximation is accurate only for small rotation angles. Yet, we show that this is not a serious restriction in our applications.

We employ our rotation in two rendering applications: (1) global illumination computation and (2) real-time shading with environment lighting. In the former case, the fast rotation is used in radiance caching for coordinate-frame alignment in radiance interpolation [Křivánek et al. 2005; Křivánek 2005]. In the latter application, the ability to perform our fast rotation in real-time on a per-pixel basis on graphics hardware is employed to enhance Kautz et al.'s shading algorithm [Kautz et al. 2002] with normal mapping. The real-time per-pixel rotation, and, consequently, the possibility to use normal maps to represent surface detail, allows us to decouple the illumination quality from the number of vertices.

The rest of this paper is organized as follows. The next section provides the background on spherical harmonic rotation; Section 3 describes our rotation approximation. Applications in radiance caching and in real-time shading are presented in Sections 4 and 5, respectively. Finally, Section 6 concludes the work.

2 Background

2.1 Spherical Harmonics

Any spherical function $L(\omega)$ can be approximated in terms of spherical harmonics as $L(\omega) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} \lambda_l^m Y_l^m(\omega)$, where ω is a direction in 3D, Y_l^m are the spherical harmonics (abbreviated SH) [Green 2003] and n is the SH approximation *order*. Coefficients λ_l^m constitute the representation of $L(\omega)$ with respect to the SH basis.

*xkrivanj@fel.cvut.cz
†jaakko@cs.ucf.edu
‡sumant@cs.ucf.edu
§kadi@irisa.fr
¶zara@fel.cvut.cz

There are n^2 coefficients in the approximation of order n. Spherical harmonics of equal index l form a *band*, with one harmonic in the first band ($m = 0$), three in the second band ($m = -1, 0, 1$), five in the third band ($m = -2, -1, 0, 1, 2$), etc. Although the coefficients have two indices l and m, they are stored in a one-dimensional array $[\lambda_0^0, \lambda_1^{-1}, \lambda_1^0, \lambda_1^1, \ldots]$, indexed by $i = l(l+1) + m$. This layout is used in the example code in this paper.

2.2 Spherical Harmonic Rotation

Problem Statement. Given a vector of SH coefficients $\Lambda = \{\lambda_l^m\}$ representing a spherical function $L(\omega) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} \lambda_l^m Y_l^m(\omega)$, find a vector of coefficients $\Upsilon = \{\upsilon_l^m\}$ representing the rotated function $L(\mathscr{R}^{-1}(\omega)) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} \upsilon_l^m Y_l^m(\omega)$, where $\mathscr{R}$ is the desired rotation.

A rotated version of any function represented by the spherical harmonics coefficients Λ of order n can exactly be represented by another set of coefficients Υ, also of order n. The rotation can be carried out as a linear transformation $\Upsilon = \mathbf{R}\Lambda$ with a block-sparse rotation matrix $\mathbf{R}$ (Figure 1). The rotation matrix $\mathbf{R}$ is composed of blocks $\mathbf{R}^l$, each corresponding to one band. Note that coefficients between different SH bands do not interact. The problem is how to construct $\mathbf{R}$ for a desired 3D rotation and order n. Different ways of solving this task are described in [Green 2003]. Our approach to SH rotation, described in Section 3, avoids explicit construction of $\mathbf{R}$. We compare our approach with the methods of Ivanic and Ruedenberg [1996; 1998] and the *zxzxz*-rotation of Kautz et al. [2002].

$$
\mathbf{R} = \left[\begin{array}{c|ccc|ccccc|c}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ \hline
0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & 0 & 0 & 0 & 0 & 0 & \cdots \\ \hline
0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \cdots \\
0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \cdots \\
0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \cdots \\
0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \cdots \\
0 & 0 & 0 & 0 & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \mathbf{X} & \cdots \\ \hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}\right]
= \begin{bmatrix}
1 & \mathbf{0} & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{R}^1 & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{0} & \mathbf{R}^2 & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{bmatrix}
$$

Figure 1: Form of the SH rotation matrix. (After [Green 2003]).

Ivanic and Ruedenberg [1996; 1998] construct $\mathbf{R}$ recurrently, starting from $\mathbf{R}^1$ continuing over $\mathbf{R}^2$ up to $\mathbf{R}^l$ for any given l. Elements of the block $\mathbf{R}^l$ are computed from elements of $\mathbf{R}^{l-1}$ and $\mathbf{R}^1$ using rules summarized in [Ivanic and Ruedenberg 1998; Green 2003]. (See Figure 1 for the composition of matrix $\mathbf{R}$ from the sub-blocks.)

A more efficient spherical harmonics rotation can be achieved with the method of Kautz et al. [2002] that we call the *zxzxz*-rotation. A general 3D rotation is first decomposed into *zyz* Euler angles (α, β, γ). The rotation around y by angle β is then expressed as a rotation around x by $\pi/2$, a rotation around z by β and a rotation around x by $-\pi/2$. The angle of the two rotations around x is fixed; therefore, the rotation matrices for them can be pre-computed. The number of non-zero elements in the x rotation matrices is only a fourth of that of a general spherical harmonics rotation matrix. Additionally, a general rotation around z is very simple (see Appendix B), therefore the *zxzxz*-rotation is more efficient than Ivanic and Ruedenberg's rotation.

Direct3D API [Microsoft 2004] provides the `D3DXSHRotate()` call that rotates a function represented by spherical harmonics. The implementation is probably based on explicit formulas for the elements of the rotation matrix in terms of Euler angles [Sloan et al. 2002], since it only works for orders up to $n = 6$. It is, furthermore, slower than the method of Ivanic and Ruedenberg [1996; 1998].

Choi et al.'s method [1999] performs the rotation in the complex domain and then converts the results back to the real domain. (Our spherical harmonics and coefficient vectors are *real*.) According to [Green 2004], this procedure is slower than the method of Ivanic and Ruedenberg [1996; 1998].

Since none of the above methods is fast enough for per-pixel rotation, we have developed the fast rotation described below.

3 Fast Rotation Approximation

This section describes our fast approximation of the spherical harmonics rotation using a truncated Taylor expansion of the rotation matrix.

According to Euler's rotation theorem, any rotation in 3D space can be described by three angles. We decompose rotations using the *zyz* convention and express them as three subsequent rotations around z, y, and z axes by angles α, β, and γ, respectively, i.e. $\mathbf{R} = \mathbf{R}_z(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(\gamma)$.

Spherical harmonics rotation around z is simple and efficient (see Appendix B). It remains to find the rotation matrix $\mathbf{R}_y(\beta)$. We replace this matrix by its truncated Taylor expansion at $\beta = 0$:

$$\mathbf{R}_y(\beta) \approx \mathbf{I} + \beta \frac{\mathrm{d}\mathbf{R}_y}{\mathrm{d}\beta}(0) + \frac{\beta^2}{2} \frac{\mathrm{d}^2\mathbf{R}_y}{\mathrm{d}\beta^2}(0),$$

where $\mathbf{I}$ is the identity matrix. Computation of the derivative matrices is described in Appendix A and they are depicted in Figure 2. The first derivative matrix $\frac{\mathrm{d}\mathbf{R}_y}{\mathrm{d}\beta}(0)$ has non-zero elements only on the superdiagonal and the subdiagonal. The elements of the second derivative matrix $\frac{\mathrm{d}^2\mathbf{R}_y}{\mathrm{d}\beta^2}(0)$ are non-zero only on the main diagonal and on the diagonal just below the subdiagonal and just above the superdiagonal. Therefore, the resulting rotation matrix approximation is very sparse. The rotation matrix $\mathbf{R}_y(\beta)$ does not have to be explicitly constructed at all because we know where the non-zero elements are.

In practice we use a so called "1.5th order" Taylor expansion, where any non-diagonal elements of the second derivative matrix are set to zero. The "1.5th order" expansion is only slightly less accurate than the second order expansion, but incurs less computation. The following C code shows how simple the y rotation is using the "1.5th order" Taylor expansion.

```
/** Rotate around y using the 1.5th order Taylor expansion
  @param beta        angle of rotation around y
*/
void shRotYdiff15(int order, float* dest, const float* src,
        const float* dySubDiag, const float* ddyDiag, float beta)
{
  float bbeta = 0.5f*beta*beta;
  dest[0] = src[0];
  for(int i=1; i<order*order-1; i++) {
    dest[i] = src[i] * (1.0 + bbeta*ddyDiag[i]) +
      beta * (dySubDiag[i]*src[i-1] - dySubDiag[i+1]*src[i+1]);
  }
  dest[i] = src[i] * (1.0 + bbeta*ddyDiag[i]) +
    beta * dySubDiag[i] * src[i-1];
}
```

$$\frac{\mathrm{d}\mathbf{R}_Y}{\mathrm{d}\beta}(0) = \begin{pmatrix} 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \frac{\mathrm{d}\mathbf{R}_Y^1}{\mathrm{d}\beta}(0) & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \frac{\mathrm{d}\mathbf{R}_Y^2}{\mathrm{d}\beta}(0) & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\mathrm{d}\mathbf{R}_Y^3}{\mathrm{d}\beta}(0) & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

where

$$\frac{\mathrm{d}\mathbf{R}_Y^1}{\mathrm{d}\beta}(0) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\frac{\mathrm{d}\mathbf{R}_Y^2}{\mathrm{d}\beta}(0) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1.73 & 0 \\ 0 & 0 & 1.73 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\frac{\mathrm{d}\mathbf{R}_Y^3}{\mathrm{d}\beta}(0) = \begin{pmatrix} 0 & 1.22 & 0 & 0 & 0 & 0 & 0 \\ -1.22 & 0 & 1.58 & 0 & 0 & 0 & 0 \\ 0 & -1.58 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2.45 & 0 & 0 \\ 0 & 0 & 0 & 2.45 & 0 & -1.58 & 0 \\ 0 & 0 & 0 & 0 & 1.58 & 0 & -1.22 \\ 0 & 0 & 0 & 0 & 0 & 1.22 & 0 \end{pmatrix}$$

$$\frac{\mathrm{d}^2\mathbf{R}_Y}{\mathrm{d}\beta^2}(0) = \begin{pmatrix} 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \frac{\mathrm{d}^2\mathbf{R}_Y^1}{\mathrm{d}\beta^2}(0) & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \frac{\mathrm{d}^2\mathbf{R}_Y^2}{\mathrm{d}\beta^2}(0) & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\mathrm{d}^2\mathbf{R}_Y^3}{\mathrm{d}\beta^2}(0) & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

where

$$\frac{\mathrm{d}^2\mathbf{R}_Y^1}{\mathrm{d}\beta^2}(0) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

$$\frac{\mathrm{d}^2\mathbf{R}_Y^2}{\mathrm{d}\beta^2}(0) = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -3 & 0 & 1.73 \\ 0 & 0 & 0 & -4 & 0 \\ 0 & 0 & 1.73 & 0 & -1 \end{pmatrix}$$

$$\frac{\mathrm{d}^2\mathbf{R}_Y^3}{\mathrm{d}\beta^2}(0) = \begin{pmatrix} -1.5 & 0 & 1.94 & 0 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 & 0 & 0 & 0 \\ 1.94 & 0 & -2.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 3.87 & 0 \\ 0 & 0 & 0 & 0 & -8.5 & 0 & 1.94 \\ 0 & 0 & 0 & 3.87 & 0 & -4 & 0 \\ 0 & 0 & 0 & 0 & 1.94 & 0 & -1.5 \end{pmatrix}$$

Figure 2: First (left) and second (right) derivative of the Y rotation matrix at $\beta = 0$. (Numbers are rounded to three significant digits.)

Arrays `dySubDiag` and `ddyDiag` contain the subdiagonal of $\frac{\mathrm{d}\mathbf{R}_y}{\mathrm{d}\beta}(0)$ and the diagonal of $\frac{\mathrm{d}^2\mathbf{R}_y}{\mathrm{d}\beta^2}(0)$, respectively. They are computed just once at the start-up and remain constant throughout run-time. The superdiagonal of $\frac{\mathrm{d}\mathbf{R}_y}{\mathrm{d}\beta}(0)$ does not have to be stored, since the first derivative matrix is, like any other infinitesimal rotation matrix, antisymmetric [Weisstein 2004].

All components for the full rotation $\mathbf{R} = \mathbf{R}_z(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(\gamma)$ are, now, available. The rotation proceeds as follows:

1. Decompose the desired rotation into the *zyz* Euler angles α, β and γ.
2. Rotate around z by α (see Appendix B) .
3. Use `shRotYdiff15()` to rotate around y by β.
4. Rotate around z by γ.

It has to be emphasized that the described procedure only *approximates* spherical harmonics rotation and is usable only if the angle of rotation around y is small. An application using our approximation must make sure that this condition holds. The next section compares the approximation error for the first, the "1.5th", and the second order Taylor expansions.

3.1 Error Analysis

Let $\mathbf{R}_y(\beta)$ be the correct matrix for rotation around y by β and let $\mathbf{R}'_y(\beta)$ be our approximation. The error caused by rotating a coefficient vector Λ using our approximation is given by the following L_2 norm:

$$\varepsilon(\beta) = \|\mathbf{R}_y(\beta)\Lambda - \mathbf{R}'_y(\beta)\Lambda\| = \|(\mathbf{R}_y(\beta) - \mathbf{R}'_y(\beta))\Lambda\| = \|\mathbf{D}(\beta)\Lambda\|.$$

Maximum of the error $\varepsilon(\beta)$ over all unit length Λ is given by the L_2 norm of the matrix $\mathbf{D}(\beta)$, which is equal to the greatest singular value of $\mathbf{D}(\beta)$. Average error $\varepsilon(\beta)$ over all unit length Λ is given by the average singular value of $\mathbf{D}(\beta)$. Figure 3 shows the maximum and the average error $\varepsilon(\beta)$ and also the actual measured $\varepsilon(\beta)$ for a Phong lobe $\cos^7(\theta)$. Although the maximum error grows quite fast with β, the results for the Phong lobe show good accuracy up to $\beta = 25^\circ$.

3.2 Complexity

In this section we compare the complexity of Ivanic and Ruedenberg's rotation [1996; 1998] and the *zxzxz*-rotation [Kautz et al. 2002] with the complexity of our approximation. The complexities are expressed in terms of order n.

Ivanic and Ruedenberg's method. The number of non-zero elements in a general spherical harmonics rotation matrix for order n is $N_{nz}(n) = \sum_{i=1}^{n}(2i-1)^2 = n(4n^2-1)/3$. Computation of each element of the matrix using Ivanic and Ruedenberg's method [1996; 1998] is a constant-time operation, therefore the complexity of the spherical harmonics rotation matrix construction is $O(n^3)$. Complexity of transforming a spherical harmonics coefficient vector with the matrix is also $O(n^3)$.

zxzxz-*Rotation.* One z-rotation involves $N_z(n) = 2n(n-1)$ multiplications; the cost of one x-rotation is $N_x(n) = \sum_{i=1}^{n}(i^2 - i + 1) = n(n^2+2)/3$. Rotation of one spherical harmonics vector with the *zxzxz*-rotation, thus, costs $3N_z(n) + 2N_x(n) = n(2n^2 + 18n - 14)/3 \in O(n^3)$ multiplications. This is only about a half of the number of multiplications needed for transforming a vector by a full spherical harmonics rotation matrix $\mathbf{R}$. Additionally, the rotation matrix itself does not have to be constructed in the *zxzxz*-rotation.

Our rotation approximation. There are $N_{\mathrm{dy}}(n) = 5n^2$ multiplications in `rotYdiff15()`. The total cost of our rotation $2N_z(n) + N_{\mathrm{dy}}(n) = 9n^2 - 4n \in O(n^2)$ is asymptotically lower than that of the previous methods. The advantage of our method in terms of speed becomes more pronounced as the order n increases, the downside being the lower accuracy for higher n.

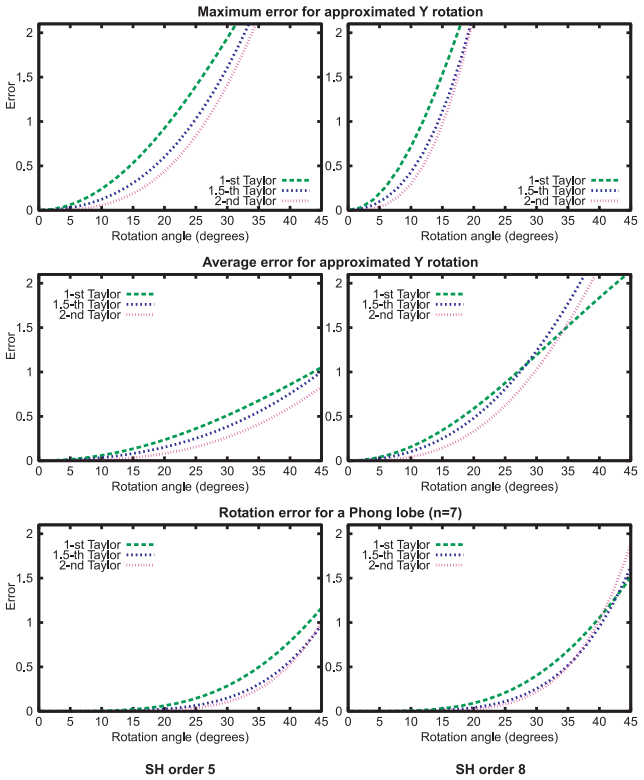


Figure 3: Rotation approximation error $\varepsilon(\beta)$ as a function of angle β for spherical harmonics of order 5 (left) and 8 (right). The plots in the first row show the maximum error for any unit length vector, the second row shows the average error over all unit length vectors, and the third row shows the actual error for a Phong lobe $\cos^7(\theta)$. Error is expressed as the Euclidean distance between coefficient vectors. Each plot shows the error for the 1st, the "1.5th", and the 2nd order Taylor expansion.

4 Application in Radiance Caching

In this section we describe the application of our fast rotation approximation in radiance caching for global illumination computation. We start by briefly reviewing the radiance caching algorithm.

4.1 Radiance Caching Overview

Radiance caching [Křivánek et al. 2005; Křivánek 2005] generalizes Ward et al.'s [1988] irradiance caching algorithm for the use on glossy surfaces.[1] It accelerates indirect illumination computation by reusing previously computed and cached incoming radiance values through interpolation over glossy surfaces.

Whenever a ray hits a glossy surface at a point $\mathbf{p}$, the radiance cache is queried for cached nearby incoming radiance values (or *records*). If no cached incoming radiance record is found near $\mathbf{p}$, radiance cache-based interpolation cannot be used. Therefore, the hemisphere above $\mathbf{p}$ is sampled by secondary rays, the sampled directional distribution of incoming radiance at $\mathbf{p}$ is projected onto spherical or hemispherical harmonics [Gautron et al. 2004], and the resulting coefficient vector Λ_i is stored with a new record in the radiance cache.

[1] Irradiance caching only supports view independent, diffuse surfaces.

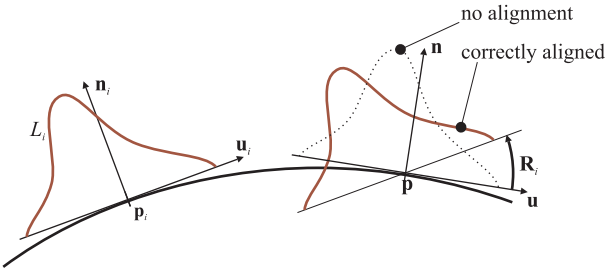


Figure 4: Rotation $\mathbf{R}_i$ aligns the coordinate frame at the record location $\mathbf{p}_i$ with the coordinate frame at the point of interpolation $\mathbf{p}$ to make interpolation possible.

If, on the other hand, cached records are found near $\mathbf{p}$, the cached incoming radiance coefficient vectors Λ_i are interpolated with the weighted average:[2]

$$\Lambda(\mathbf{p}) = \frac{\sum_{\mathscr{S}} (\mathbf{R}_i \Lambda_i) w_i(\mathbf{p})}{\sum_{\mathscr{S}} w_i(\mathbf{p})}. \quad (1)$$

Weight of the i-th cache record with respect to point $\mathbf{p}$ is given by $w_i(\mathbf{p}) = \left(\frac{\|\mathbf{p}-\mathbf{p}_i\|}{R_i} + \sqrt{1-\mathbf{n}\cdot\mathbf{n}_i} \right)^{-1}$, where $\mathbf{n}$ is the normal at the point of interpolation $\mathbf{p}$, $\mathbf{p}_i$ is the location of the i-th cache record, and $\mathbf{n}_i$ is the normal at $\mathbf{p}_i$. The harmonic mean distance, R_i, to the objects visible from $\mathbf{p}_i$, is computed from the ray lengths during hemisphere sampling and stored in the cache. The set $\mathscr{S}$ of records used for interpolation is defined as $\mathscr{S} = \{i | w_i(\mathbf{p}) > 1/a\}$, where a is a user defined maximum allowed interpolation error.

The important thing here is the rotation $\mathbf{R}_i$ that has to be used to align the coordinate frames at the point of interpolation $\mathbf{p}$ and the record location $\mathbf{p}_i$: the cached incoming radiance *has to be rotated* before the interpolation is possible (Figure 4). This means that there is one or more rotations per interpolation.

4.2 The Use of our Rotation in Radiance Caching

Due to the interpolation criterion based on the weight $w_i(\mathbf{p})$, the normals at the record location $\mathbf{p}_i$ and the point of interpolation $\mathbf{p}$ are always similar (if they were not, the record at $\mathbf{p}_i$ would not be used for interpolation at $\mathbf{p}$, because of the $\sqrt{1-\mathbf{n}\cdot\mathbf{n}_i}$ term in the definition of $w_i(\mathbf{p})$). The angle of rotation around the y-axis in the Euler zyz decomposition of $\mathbf{R}_i$ corresponds to the angle between the two normals $\mathbf{n}_i$ and $\mathbf{n}$, hence it is always small. We can therefore safely apply our rotation approximation for the coordinate frame alignment.

Rotation Approximation Limiting Angle. To keep the error caused by the rotation approximation low, we use the approximation only for angles $\beta = \angle(\mathbf{n}, \mathbf{n}_i)$ smaller than a threshold β_{lim}. The accurate, but more costly, $zxzxz$-rotation is used if $\beta > \beta_{\text{lim}}$. We set the limiting angle to $\beta_{\text{lim}} = 1.25a$ (derivation in Appendix C), where a is the user specified maximum allowed interpolation error of radiance caching. This allows higher error in rotation if the user allows higher error in interpolation. Additionally, setting β_{lim} to a multiple of a has for consequence that the percentage of $zxzxz$-rotations in a given scene is constant, no matter what value the user specifies for a.

[2] Only a simplified version of the actual formula, omitting translational gradients, is shown here.

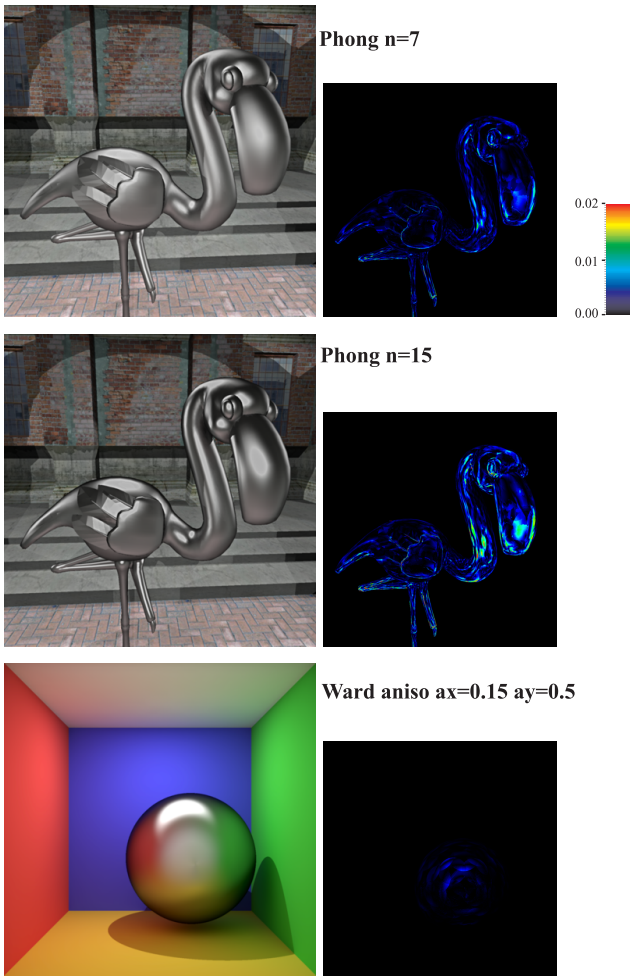


Figure 5: Left: Radiance caching renderings obtained with our approximate rotation. The flamingo is assigned a Phong BRDF with the exponent of 7 (top) or 15 (middle) and the glossy sphere has an anisotropic Ward BRDF [Ward 1992] with $\alpha_x = 0.15$, $\alpha_y = 0.5$. Spherical harmonics order $n = 10$ is used for all renderings. Right: Color coded difference between images with approximated and correct rotation, measured on a $[0,1]$ RGB scale. The difference is below the visual threshold of 1% for most pixels.

If, on the other hand, we kept β_{lim} independent of the caching error a, increasing a would lead to a more frequent use of the *zxzxz*-rotation. This would actually slow down the interpolation—an unexpected effect of increasing the allowed caching error.

Our rotation approximation and the *zxzxz*-rotation may not meet in a visually continuous fashion at β_{lim} if a is high ($a > 0.3$ in our scenes). However, in such a case, the caching artifacts are more serious anyway and the rotation artifacts go unnoticed.

4.3 Radiance Caching Results

For the results in this section we used the "1.5th Taylor expansion" of the y rotation matrix (all non diagonal elements of the second derivative matrix are ignored). In Figure 5 we compare the results of radiance caching obtained by the correct and the approximated rotation. Instead of a side-by-side comparison, in which the results are visually indistinguishable, we show a color coded difference between the results of the two methods. Image areas exhibiting the maximum error are usually very curved. In those areas, visual artifacts, if any, are well-masked.

Table 1 shows the rendering times for the flamingo and the sphere (Figure 5 left) with radiance caching. The rendering time only includes the interpolation from the cache. For SH order $n = 6$, our method is 4 times faster than the *zxzxz*-rotation; for $n = 10$ it is 6 times faster.

Order	6		10	
	RT	TPR	RT	TPR
Flamingo				
Ignore	10.3 s	—	11.2 s	—
Our method	**12.8 s**	**0.68 μs**	**16.9 s**	**1.54 μs**
zxzxz	21.2 s	2.96 μs	47.4 s	9.83 μs
Ivanic	47.3 s	10.1 μs	192 s	49.1 μs
DirectX	76.4 s	17.9 μs	—	—
Sphere				
Ignore	3.30 s	—	3.96 s	—
Our method	**4.28 s**	**0.65 μs**	**6.13 s**	**1.44 μs**
zxzxz	7.08 s	2.51 μs	16.8 s	8.57 μs
Ivanic	17.8 s	9.63 μs	75.8 s	47.8 μs
DirectX	30.3 s	17.9 μs	—	—

Table 1: Rendering times for the flamingo and sphere images in Figure 5 with radiance caching. The rendering time includes only interpolation from the cache. Various rotation methods are used for interpolation: Ignore (rotation is ignored), our method, *zxzxz*-rotation, the method of Ivanic and Ruedenberg and the DirectX rotation. 'RT' is the frame rendering time and 'TPR' is the time per rotation. There were $1,226,917 \times 3 = 3,680,751$ rotations for the flamingo and $501,420 \times 3 = 1,504,260$ rotations for the sphere.

5 Application in Real-time Shading

To demonstrate the use of our fast rotation approximation in real-time shading, we extend the rendering method of Kautz et al. [2002] to compute shading on objects with a per-pixel modulated normal. We start by giving a short overview of the original algorithm.

5.1 Fast, Arbitrary BRDF Shading for Low-Frequency Lighting

Kautz et al. [2002] uses graphics hardware to perform real-time shading of surfaces with arbitrary BRDFs illuminated by low-frequency environment lighting. By environment lighting we understand the light incident at an object from the whole sphere of directions. Shadowing is ignored and an assumption is made that the lighting is spatially invariant. Therefore, each point on an object surface receives the same illumination.

Spherical harmonics are used to represent both the environment lighting and BRDFs. A BRDF is represented as a 2D table, whose each element stores the spherical harmonics coefficient vector of the BRDF lobe for one fixed outgoing (viewing) direction. The lighting integral for a given viewing direction is computed as a dot product of the environment lighting coefficients and the BRDF coefficients for that outgoing direction. The rendering algorithm proceeds as follows:

Figure 6: Detail of a normal mapped vase rendered with our SH rotation (right) and with the simplified normal mapping (left). Normal mapping with our SH rotation is more successful at conveying the shape approximated by the normal map. The vase is illuminated by the St Peter's Basilica environment map; the BRDF comes from a measurement of a brushed metal [cite westin00lafortune].

1. [Per-vertex, CPU] Rotate the lighting coefficients to the local coordinate frame of vertex v_p. Send the rotated coefficient as vertex data to the GPU.
2. [Per-pixel, GPU] Look up the BRDF coefficients for the viewing direction transformed to the local coordinate frame.
3. [Per-pixel, GPU] Compute the dot product of the BRDF coefficients and the rotated lighting coefficients.

In this technique, the shading variance due to surface orientation is limited by the number of vertices in the mesh. A very fine mesh must be used for detailed shading. Our extension of the algorithm, presented in the next section, removes this restriction by allowing to modulate the surface normal by a texture (normal mapping).

5.2 Shading on Normal-Mapped Surfaces through a Per-Pixel Rotation

Normal maps modulate surface normals by a texture in order to represent small surface variations, for which an explicit geometry representation would be too bulky. Our extension of the shading algorithm of Kautz et al. allows to use normal mapping on an object illuminated by environment lighting, thereby decoupling the possible shading details from the number of mesh vertices.

We modify the original rendering algorithm in the following way (new steps are in italics):

1. [Per-vertex, CPU] Rotate the lighting coefficients to the local coordinate frame of vertex v_p. Send the rotated coefficient as vertex data to the GPU.
2. [Per-pixel, GPU] *Look up the normal map (normal map represents the modulation of the local coordinate frame at the pixel with respect to the frame given by the interpolated per-vertex normals).*
3. [Per-pixel, GPU] Look up the BRDF coefficients for the viewing direction transformed to the *modulated* local coordinate frame.
4. [Per-pixel, GPU] *Use our fast spherical harmonics rotation approximation to rotate the BRDF coefficients from the modulated local frame to the interpolated per-vertex local frame.*
5. [Per-pixel, GPU] Compute the dot product of local lighting and BRDF coefficients.

Modulation of surface normals by a normal map is usually limited to rather small angles; we can therefore safely use our rotation approximation. Thanks to the approximation simplicity, we were able to implement the per-pixel rotation in a pixel shader of the graphics hardware.

Our extension leads to a substantial improvement of visual quality as illustrated in Figures 6 and 7. It also allows using meshes with lower number of vertices than the original algorithm, which improves the overall rendering performance.

Simplified Normal Mapping. To simplify normal mapping, one can ignore the per-pixel rotation (step 4) and use the normal map only to modulate the local frame for the BRDF lookup. Unlike our method, this simplified normal mapping generates flat looking surfaces and does not capture color variations on the surface bumps due to multicolored environment lighting.

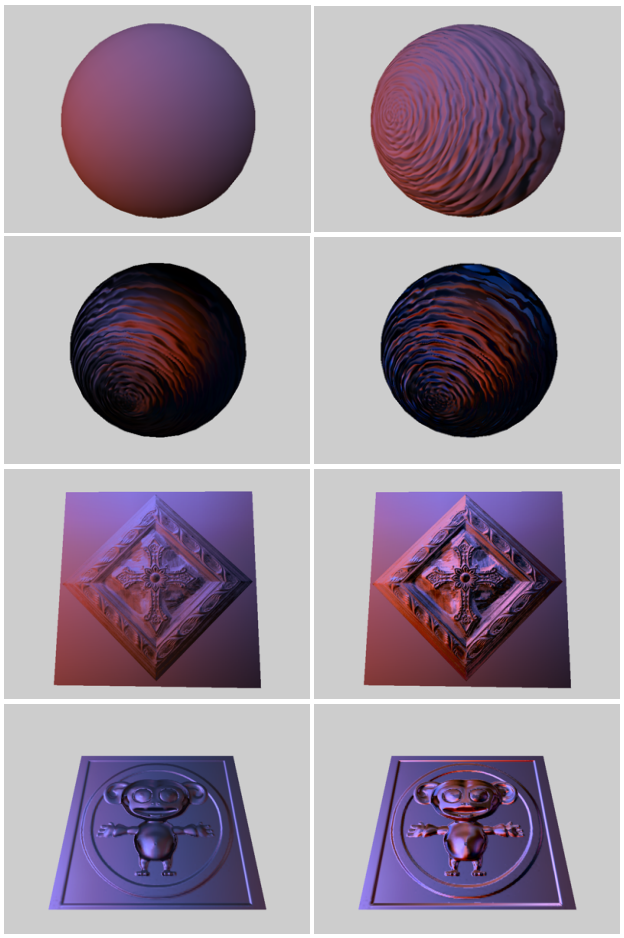
5.3 Results

Figures 6 and 7 compare the results obtained with our rotation to the results of the simplified normal mapping. We used spherical harmonics of order $n = 5$ (25 coefficients). The rotation approximation used the "1.5th order" Taylor expansion for bands $l = 1$ to $l = 3$ and the first order Taylor expansion for band $l = 4$. Due to the limited pixel-shader instruction-count we had to use four passes to accommodate 25 coefficients. The frame rates for these images at resolution 800×600 were:

	Simplified	Rotation
Vase (891 vertices)	58 fps	46 fps
Sphere (560 vertices)	65 fps	54 fps
Plane (25 vertices)	75 fps	56 fps

These figures were measured on a 2.2 GHz Pentium IV PC with ATI Radeon 9800 Pro GPU. The drop in the frame rate due to the rotation is more pronounced for low polygon count meshes, for which the rendering time is determined mostly by fragment processing.

Although those results are outdated by the power of recent graphics hardware, the presented algorithm is useful as a demonstration of the possible uses of our fast rotation approximation.



Simplified Our rotation

Figure 7: More results of the normal mapping with our SH rotation (right) compared to the simplified normal mapping (left). The BRDFs used were (from top to bottom) Lambertian, Phong, Ward isotropic, Ward anisotropic. The objects are illuminated by the Grace Cathedral environment map. Note the color variations on the surface bumps captured by our method. For the Lambertian surface (top), the simplified normal mapping does not work since the BRDF is view independent. Our method is also more successful at revealing the effects of BRDF anisotropy (bottom).

6 Conclusion

We presented a fast algorithm for rotating functions represented by spherical harmonics. The main idea is to replace a general spherical harmonics rotation matrix by its truncated Taylor expansion. The algorithm has lower complexity and is faster than previous approaches. Although the rotation approximation is accurate only for small rotation angles, we have demonstrated its practical usefulness in global illumination computation (radiance caching) and real-time rendering. The rotation algorithm is simple enough to fit in the pixel shader of standard graphics hardware, which allows to apply the rotation on a per-pixel basis in real-time. We took advantage of this by shading normal mapped surfaces with arbitrary BRDFs illuminated by environment lighting.

We believe that an approach similar to that presented in this paper could be used to approximate rotation of functions represented by wavelets, which is the main avenue of future work. Our fast rotation can also be used to extend normal mapping in pre-computed radiance transfer [Sloan 2006] to view-dependent BRDFs.

Acknowledgements

This work was supported by France Telecom R&D, Rennes, France and by the Ministry of Education of the Czech Republic under the the research program LC-06008 (Center for Computer Graphics).

References

CHOI, C. H., IVANIC, J., GORDON, M. S., AND RUEDENBERG, K. 1999. Rapid and stable determination of rotation matrices between spherical harmonics by direct recursion. *J. Chem. Phys. 111*, 19, 8825–8831.

GAUTRON, P., KŘIVÁNEK, J., PATTANAIK, S. N., AND BOUATOUCH, K. 2004. A novel hemispherical basis for accurate and efficient rendering. In *Rendering Techniques 2004, Eurographics Symposium on Rendering*, 321–330.

GREEN, P., KAUTZ, J., MATUSIK, W., AND DURAND, F. 2006. View-dependent precomputed light transport using nonlinear gaussian function approximations. In *Proceedings of ACM 2006 Symposium in Interactive 3D Graphics and Games*.

GREEN, R. 2003. Spherical harmonic lighting: The gritty details. In *Game Developpers' Conference*.

GREEN, R., 2004. Personal communication.

IVANIC, J., AND RUEDENBERG, K. 1996. Rotation matrices for real spherical harmonics. direct determination by recursion. *J. Phys. Chem. 100*, 15, 6342–6347.

IVANIC, J., AND RUEDENBERG, K. 1998. Additions and corrections : Rotation matrices for real spherical harmonics. *J. Phys. Chem. A 102*, 45, 9099–9100.

KAUTZ, J., SLOAN, P.-P., AND SNYDER, J. 2002. Fast, arbitrary BRDF shading for low-frequency lighting using spherical harmonics. In *Proceedings of the 13th Eurographics workshop on Rendering*, Eurographics Association, 291–296.

KŘIVÁNEK, J., GAUTRON, P., PATTANAIK, S., AND BOUATOUCH, K. 2005. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics (Accepted for publication)*.

KŘIVÁNEK, J. 2005. *Radiance Caching for Global Illumination Computation on Glossy Surfaces*. PhD thesis, Université de Rennes 1 and Czech Technical University in Prague.

LEHTINEN, J., AND KAUTZ, J. 2003. Matrix radiance transfer. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, ACM Press, 59–64.

LIU, X., SLOAN, P.-P., SHUM, H.-Y., AND SNYDER, J. 2004. All-frequency precomputed radiance transfer for glossy objects. In *Proceedings of the Eurographics Symposium on Rendering*, 337–344.

MICROSOFT, 2004. Directx 9.0 SDK update, October.

NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. 2004. Triple product wavelet integrals for all-frequency relighting. *ACM Trans. Graph. 23*, 3, 477–487.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, 527–536.

SLOAN, P.-P., HALL, J., HART, J., AND SNYDER, J. 2003. Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph. 22*, 3, 382–391.

SLOAN, P.-P., LUNA, B., AND SNYDER, J. 2005. Local, deformable precomputed radiance transfer. *ACM Trans. Graph. 24*, 3, 1216–1223.

SLOAN, P.-P. 2006. Normal mapping for precomputed radiance transfer. In *Proceedings of ACM 2006 Symposium in Interactive 3D Graphics and Games*.

WANG, R., TRAN, J., AND LUEBKE, D. 2004. All-frequency relighting of non-diffuse objects using separable BRDF approximation. In *Proceedings of the Eurographics Symposium on Rendering*, 345–354.

WANG, R., TRAN, J., AND LUEBKE, D. 2005. All-frequency interactive relighting of translucent objects with single and multiple scattering. *ACM Trans. Graph. 24*, 3, 1208–1215.

WARD, G. J., RUBINSTEIN, F. M., AND CLEAR, R. D. 1988. A ray tracing solution for diffuse interreflection. In *Proceedings of SIGGRAPH '88*, 85–92.

WARD, G. J. 1992. Measuring and modeling anisotropic reflection. In *Proceedings of SIGGRAPH*, ACM Press, 265–272.

WEISSTEIN, E. W., 2004. Infinitesimal rotation. From *MathWorld*–A Wolfram Web Resource. http://mathworld.wolfram.com/InfinitesimalRotation.html.

WESTIN, S. H. Lafortune BRDF for RenderMan. http://www.graphics.cornell.edu/ westin/lafortune/lafortune.html.

A SH Rotation Matrix Derivative

Here we describe the computation of the k-th derivative matrix $\frac{\mathrm{d}^k \mathbf{R}_Y}{\mathrm{d}\beta^k}$. The algorithm is based on Ivanic and Ruedenberg's rotation matrix construction [1996; 1998] and retains its structure. Elements of the derivative matrix block $\frac{\mathrm{d}^k \mathbf{R}_Y^l}{\mathrm{d}\beta^k}$ are indexed by m_1 and m_2 and we denote them $R_Y^{(k)}(l,m_1,m_2)$. We start with bands $l=0$ and $l=1$:

$$
\begin{array}{llll}
R_Y^{(k)}(0,\ 0,\ 0) = 1^{(k)} & R_Y^{(k)}(1,\ 0,\ 0) = \cos^{(k)}(\beta) \\
R_Y^{(k)}(1,\ -1,\ -1) = 1^{(k)} & R_Y^{(k)}(1,\ 0,\ 1) = -\sin^{(k)}(\beta) \\
R_Y^{(k)}(1,\ -1,\ 0) = 0 & R_Y^{(k)}(1,\ 1,\ -1) = 0 \\
R_Y^{(k)}(1,\ -1,\ 1) = 0 & R_Y^{(k)}(1,\ 1,\ 0) = \sin^{(k)}(\beta) \\
R_Y^{(k)}(1,\ 0,\ -1) = 0 & R_Y^{(k)}(1,\ 1,\ 1) = \cos^{(k)}(\beta)
\end{array}
$$

where $\beta = 0$ and $1^{(k)}$ is the derivative of one ($1^{(k)} = 1$ if $k = 0$ and $1^{(k)} = 0$ if $k > 0$). For higher bands, we compute simultaneously the zero-th, first, second, ... *maxderiv*-th derivative:

for $l = 2 \ldots n-1$ **do**
  for $k = 0 \ldots maxderiv$ **do**
    for $m_1 = -l \ldots l$ **do**
      for $m_2 = -l \ldots l$ **do**

$$
\begin{aligned}
R^{(k)}(l,m_1,m_2) \quad := \quad & u^l_{m_1 m_2} \cdot dU^{(k)}(l,m_1,m_2) + \\
& v^l_{m_1 m_2} \cdot dV^{(k)}(l,m_1,m_2) + \\
& w^l_{m_1 m_2} \cdot dW^{(k)}(l,m_1,m_2)
\end{aligned}
$$

      end for
    end for
  end for
end for

Numerical coefficients $u^l_{m_1 m_2}$, $v^l_{m_1 m_2}$ and $w^l_{m_1 m_2}$ are the same as in the original paper [Ivanic and Ruedenberg 1998]. Functions dU, dV and dW are defined as:

$$
dU^{(k)}(l,m_1,m_2) = dP^{(k)}(l,m_1,m_2,0)
$$

$$
dV^{(k)}(l,m_1,m_2) = \begin{cases}
dP^{(k)}(l,m_1-1,m_2,1) - dP^{(k)}(l,-m_1+1,m_2,-1) & \text{if } m_1 > 1 \\
\sqrt{2}\,dP^{(k)}(l,0,m_2,1) & \text{if } m_1 = 1 \\
dP^{(k)}(l,1,m_2,1) + dP^{(k)}(l,-1,m_2,-1) & \text{if } m_1 = 0 \\
\sqrt{2}\,dP^{(k)}(l,0,m_2,-1) & \text{if } m_1 = -1 \\
dP^{(k)}(l,-m_1-1,m_2,-1) + dP^{(k)}(l,m_1+1,m_2,1) & \text{if } m_1 < -1
\end{cases}
$$

$$
dW^{(k)}(l,m_1,m_2) = \begin{cases}
dP^{(k)}(l,m_1+1,m_2,1) + dP^{(k)}(l,-m_1-1,m_2,-1) & \text{if } m_1 > 0 \\
dP^{(k)}(l,m_1-1,m_2,1) - dP^{(k)}(l,-m_1+1,m_2,-1) & \text{otherwise}
\end{cases}
$$

with

$$
dP^{(k)}(l,m_1,m_2,i) = \begin{cases}
dT^{(k)}(1,i,0,l-1,m_1,m_2) & \text{if } |m_2| < l \\
dT^{(k)}(1,i,1,l-1,m_1,l-1) - dT^{(k)}(1,i,-1,l-1,m_1,-l+1) & \text{if } m_2 = l \\
dT^{(k)}(1,i,1,l-1,m_1,-l+1) + dT^{(k)}(1,i,-1,l-1,m_1,l-1) & \text{if } m_2 = -l
\end{cases}
$$

and

$$
dT^{(k)}(l,m_1,m_2,l',m_1',m_2') = \sum_{i=0}^{k} \binom{k}{i} R_y^{(i)}(l,m_1,m_2) \cdot R_y^{(k-i)}(l',m_1',m_2').
$$

Function dT implements the product derivative rule $(fg)^{(k)} = \sum_{i=0}^{k} \binom{k}{i} f^{(i)} g^{(k-i)}$, where $f^{(k)}$ denotes the k-th derivative.

B SH Rotation around z-Axis

The Z-rotation is computed efficiently without constructing the rotation matrix $\mathbf{R}_Z(\alpha)$ using the following procedure:

for $l = 0 \ldots n-1$ **do**
  $v_l^0 := \lambda_l^0$
  for $m = 1 \ldots l$ **do**
    $v_l^{-m} := \lambda_l^{-m} \cos(m\alpha) - \lambda_l^m \sin(m\alpha)$
    $v_l^m \ := \lambda_l^{-m} \sin(m\alpha) + \lambda_l^m \cos(m\alpha)$
  end for
end for

The sines and cosine of multiple angles can be computed with the recurrence formula:

$$
\begin{aligned}
\sin(m\alpha) &= 2\sin((m-1)\alpha)\cos(\alpha) - \sin((m-2)\alpha) \\
\cos(m\alpha) &= 2\cos((m-1)\alpha)\cos(\alpha) - \cos((m-2)\alpha)
\end{aligned}
$$

The number of multiplications in the rotation procedure is $N_Z(n) = 2n(n-1)$ thus the complexity is $O(n^2)$.

C Derivation of the Limiting Angle

This section justifies setting the limiting angle β_{lim} in radiance caching to a multiple of the maximum caching error a (see Section 4.2). A radiance cache record i is included in interpolation at a point $\mathbf{p}$ if $a > 1/w_i(\mathbf{p})$. Weight $w_i(\mathbf{p}) = \left(\frac{\|\mathbf{p}-\mathbf{p}_i\|}{R_i} + \sqrt{1-\mathbf{n}\cdot\mathbf{n}_i} \right)^{-1}$ is a function of the distance $\|\mathbf{p}-\mathbf{p}_i\|$ and the angle between normals $\beta = \angle(\mathbf{n},\mathbf{n}_i)$. Consider a constant curvature surface with the osculating circle radius r. Then $\|\mathbf{p}-\mathbf{p}_i\| = 2r\sin\frac{\beta}{2}$ and therefore the weight is only a function of β, *i.e.*

$$
1/w_i(\mathbf{p}) = f(\beta) = \frac{2r}{R_i}\sin\frac{\beta}{2} + \sqrt{1-\cos\beta},
$$

where R_i is the harmonic mean distance.

Our aim is to find for a given a and a fixed r the value of β_{lim} such that for all accepted radiance cache records, the normal divergence is never more than β_{lim}. This can be done by inverting function f, which is unfortunately impossible to do in a closed form. Instead we take the first order Taylor expansion of f at $\beta = 0$, which is $f(\beta) \approx \beta(1/\sqrt{2} + \frac{r}{R_i})$, and we find $\beta_{\text{lim}} = a(1/\sqrt{2} + \frac{r}{R_i})^{-1}$. We choose $\frac{r}{R_i} = 0.1$ and get $\beta_{\text{lim}} = 1.25a$ (in degrees $\beta_{\text{lim}} = 70.1a$). The choice $\frac{r}{R_i} = 0.1$ means that all surfaces of curvature $10/R_i$ or smaller will be rendered without exceeding β_{lim}. To conclude, by setting $\beta_{\text{lim}} = 1.25a$, the slower *zxzxz*-rotation will be used only on surfaces of curvature higher than $10/R_i$, regardless of the value of a.