Spherical Function Representations: a Practical Survey

Johannes Jendersie
TU-Clausthal

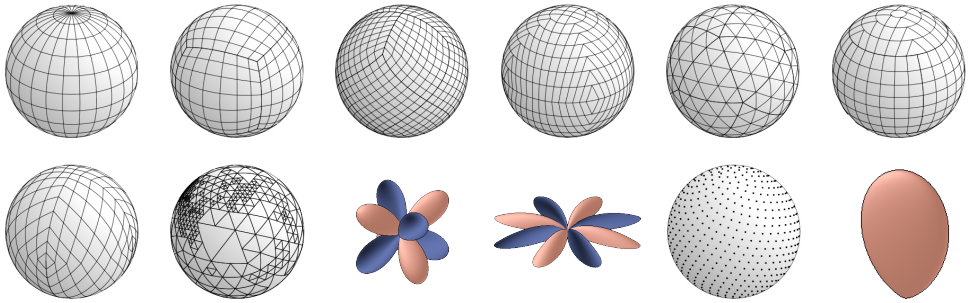


Figure 1. Examples of spherical function representations. Top row: Polar map, cube map, HEALPix, hemispherical projection, icosahedral map and Isocube map. Bottom row: Octahedral map, wavelet tree, spherical harmonic, Zernike's basis, a center distribution for mixture models and an anisotropic cosine lobe.

Abstract

In many simulation processes, storing spherical information is necessary. Particularly, we were interested which representation is the best to store light and surface descriptions in light transport simulation. This survey summarizes many different spherical mapping methods, polynomial bases and mixture models to store spherical information. For each we give practical implementation hints and show advantages as well as disadvantages. We found that mixture models outperform all other representations and evaluate three different approaches to fit them. Finally, we measure the compression performance of all models against different data sets.

1. Introduction

The need to store functions in a spherical domain appears in a broad range of topics. Among them are the storage of geographic data, simulations of earth systems like the climate and the description of physical properties. They are also frequently used

in computer graphics for light transport simulations. These require information like incident or excident light, normal distributions or reflection distributions. Latter is not only a function of a single direction but a function of two directions.

In any case there is no natural mapping from the sphere to a 2D plane and therefore it is difficult to map a spherical function into memory. We analyze the existing models to store such functions and evaluate how they can be fitted fast and stable. Thereby, we focus on extreme compression down to a few bytes mainly for the application in computer graphics.

Traditionally, cube maps (see Sec. 3.2) and spherical harmonics (see Sec. 4.2) are used for algorithms in graphics. Cube maps are used for environment maps and reflection probes [Greene 1986] with medium to high resolutions. Spherical harmonics are used whenever many heavily compressed functions must be stored. The most known application ares Precomputed Radiance Transfer (PRT) algorithms [Sloan et al. 2002], irradiance [Greger et al. 1998] and radiance caches [Krivanek et al. 2005; Vardis et al. 2014] where light distributions are approximated. They are also used for cheap sky lighting instead of (cube) environment maps, as integration can be performed by a single dot product. Another use case are normal distributions for the modeling of reflection distributions as in [Crassin et al. 2011; Heitz et al. 2015]. A related survey [Cigolle et al. 2014] details the compression of a single normal. This involves a 2D parametrization of the direction vector, which is very similar to storing distributions in a grid using the same parametrization. We were interested in the difference between the most used forms and how far alternatives can yield even better results.

A large section of this document will dive into (Gaussian) mixture models. Those would perform better in many of the mentioned applications, but are inherently difficult to fit to a target function. Only few parameters can be determined by linear optimization. All others must be found by non-linear optimization via Levenberg-Marquardt [Levenberg 1944; Marquardt 1963] or similar algorithms. Another option is the iterative *Expectation Maximization* (EM) algorithm [Dempster et al. 1977] and a third the mixture reduction techniques [Runnalls 2007; Crouse et al. 2011; Ardeshiri et al. 2015]. We compare all three against each other in the hope that they will find a way into new application fields.

Another very important usage is in the form of general Bidirectional Reflection and Scattering Distribution Functions (BRDFs and BSDFs). There are many analytic models for specific surfaces for which Montes et al. [2012] give a good overview. However, we are also interested in general compressing methods to approximate the reflectance of more complex (macroscopic) objects. Those BRDFs were briefly surveyed by Rusinkiewicz [1997] and Kurt et al. [2009], but both do not analyze the performance of the models. We show two approaches how to use the spherical functions in the bidirectional domain, but do not explore the vast number of resulting combinations. It can be expected that the properties of the spherical functions are

inherited from the two-dimensional domain.

We analyze function representations on the sphere $\mathbb{S}^2$ and the hemisphere with respect to the accuracy, the implementation effort and the time complexity of the fitting. Our review includes several different grid models, wavelet compression, spherical polynomial bases, lobe mixture models and spherical radial basis functions. Further, we compare different fitting methods in case of mixture models. Our contributions are:

- An overview of many different bases for spherical functions
- A test data set for normal distribution functions
- A generalization of the $\mathcal{H}$ Basis from Habel and Wimmer [2010], as well as modifications to Makhotkin's basis [1996]
- *Golden Ratio* low-discrepancy series on the sphere with a better distribution than a Hammersley point set.
- New parameter estimations for the EM algorithm [Dempster et al. 1977] to fit numerous isotropic and anisotropic kernels.
- A fitting algorithm for mixture models which combines Runnalls' reduction technique [2007] and EM

2. Evaluation Setup

In general a good representation is one which approximates a given function with as little as possible cost with respect to memory and/or runtime. Unfortunately, there is no natural mapping from surface of the sphere into memory. Thus, a mapping can either be area preserving (equal area) or angle preserving (conformal). For the general application of function representation the equal area property is of higher importance. It guarantees that single data values represent the same solid angle and information is uniformly preserved. However, in some cases the data might have a specific structure which benefits from a locally increased information density. Further, there are non-mapping based representations like polynomial bases and mixture models for which properties like equal area or conformal do not apply. To compare all those models, we analyze different error metrics over sets of different data. More specific properties like area preservation are given in the respective sections of the models where meaningful.

2.1. Test Data Base

Evaluating the performance of different models statistically requires a broad range of data sets. We used two different types of input data which are likely use cases for the usage of spherical functions in computer graphics.

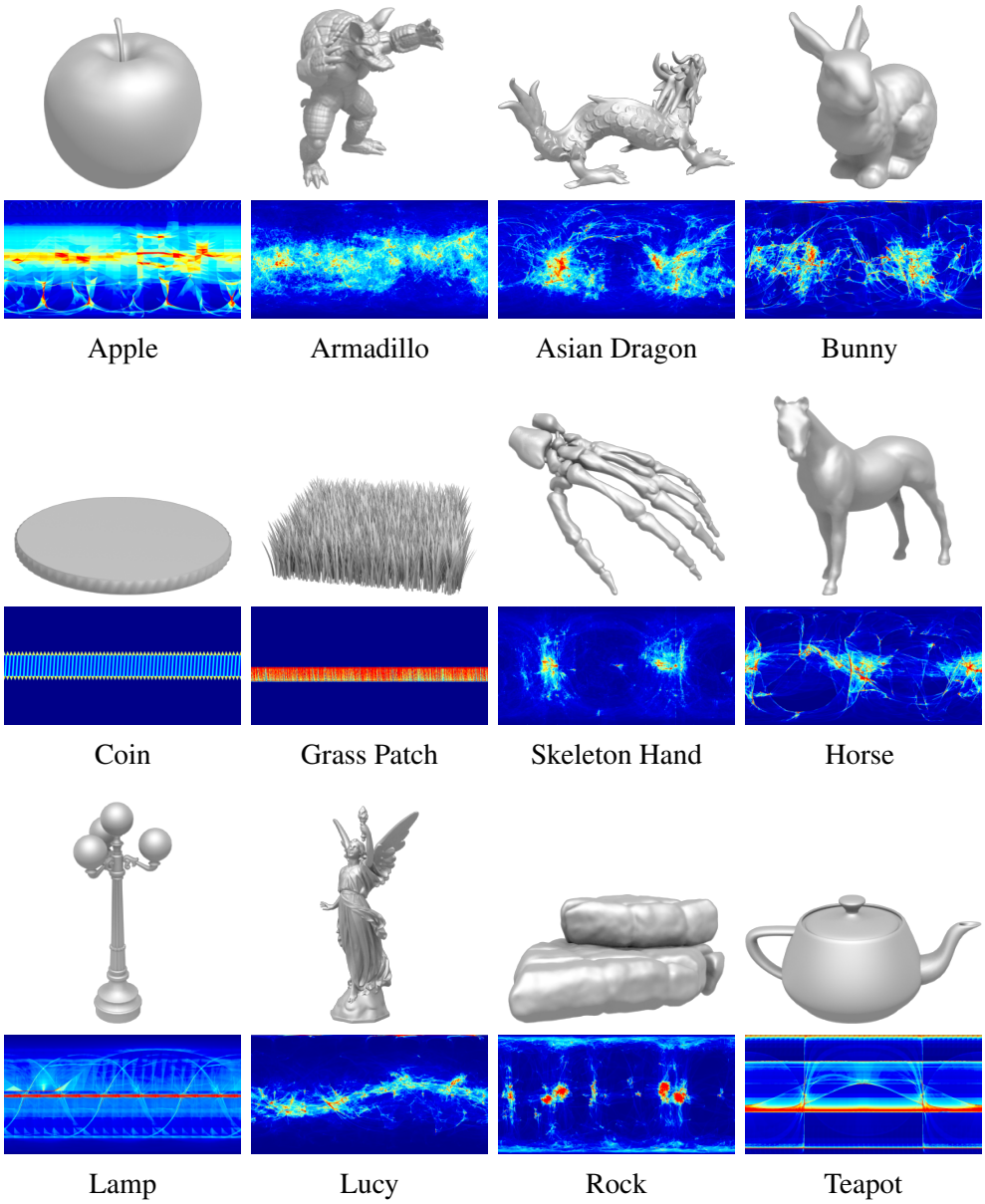


Figure 2. The Normal Distribution Function (NDF) test data set. The renderings show the objects from which the distributions were created. Below, the NDFs are previewed in polar coordinates. The actual data is stored in cube maps and not shown here.

Our first two data sets are collections of environment maps (cube maps). Often hemispherical or spherical functions are used to describe light probes generated from environment maps. We use 12 LDR maps from Emil Persson (aka Humus) [Persson] and 12 HDR maps from Paul Debevec [Debevec] / HDRI [Zaal]. The resolution of the cube maps is $6 \cdot 2048^2$ for the LDR and $6 \cdot 256^2/6 \cdot 1024^2$ for the HDR set. The

sets will be referred as LDRdat and HDRdat respectively.

Our second type of data are normal distribution functions (NDFs). We generated those distributions from the different 3D models shown in Figure 2. The NDFs are stored in $6 \cdot 1024^2$ cube maps to match the other data sets in the input format. They are normalized to yield one over the spherical integral. The generated cube maps are provided with the supplemental materials. This data set will be referred as NDFdat.

2.2. Error Measurement

For the evaluation of fitting quality we used different error measurement functions. Our first choice was *Root Mean Square Error* (RMSE), also known as L_2 error. However, this measurement is not necessarily equal to the perceived error. Discontinuity artifacts are as bad as a little increased brightness with respect to that error. Therefore, we also used the *Structural Similarity* (SSIM) Measure from Wang et al. [2004].

In case of RMSE we sample the error with the set $\mathcal{S}$ of directions at pixel centers of the original data:

$$\operatorname{rmse}(f, f') = \sqrt{\frac{1}{|\mathcal{S}|} \sum_{\vec{s} \in \mathcal{S}} (f(\vec{s}) - f'(\vec{s}))^2}$$

where f is the input function and f' the fitted result.

The same set is used for the SIM index

$$\operatorname{ssim}(f, f') = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \frac{(2\mu_f(s)\mu_{f'}(s) + c_1)(2\sigma_{ff'}(s) + c_2)}{(\mu_f^2(s) + \mu_{f'}^2(s) + c_1)(\sigma_f^2(s) + \sigma_{f'}^2(s) + c_2)}$$

which requires local mean μ, standard deviation σ and correlation $\sigma_{ff'}$ to be computed. Therefore, we sample the surrounding of s with directions $o_i(s)$ and use a Gaussian kernel for weighting.

$$\begin{aligned}
w_i(s) &= e^{-(c_3 \arccos(o_i(s) \cdot s))^2} \\
\mu(s) &= \frac{\sum_{i=1}^N w_i f(o_i(s))}{\sum_{i=1}^N w_i} \\
\sigma(s) &= \sqrt{\frac{\sum_{i=1}^N w_i (f(o_i(s)) - \mu(s))^2}{\sum_{i=1}^N w_i}} \\
\sigma_{ff'}(s) &= \frac{\sum_{i=1}^N w_i (f(o_i(s)) - \mu_f(s))(f'(o_i(s)) - \mu_{f'}(s))}{\sum_{i=1}^N w_i}
\end{aligned}$$

We used 32 directions $o_i(s)$ sampled in a cone of $2\,\mathrm{deg}$ total opening angle. For the constants we took $c_1 = 0.02$, $c_2 = 0.018$ and $c_3 = 100$ for all experiments.

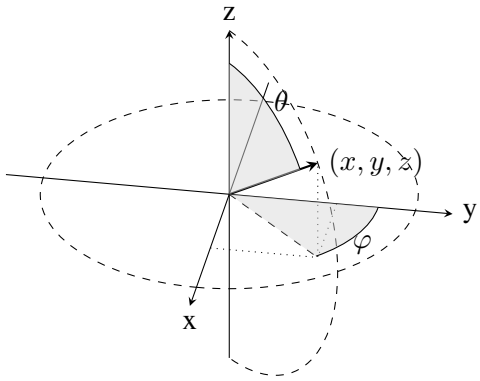
Finally, we introduce *Area Deviation* to quantify the quality of mapping based approaches as

$$A \pm \left[\lim_{n \to \infty} \frac{\omega_{\min}(n)}{\bar{\omega}}, \lim_{n \to \infty} \frac{\omega_{\max}(n)}{\bar{\omega}} \right],$$

where $\bar{\omega}$ is the average solid angle ($4\pi/n$ on the sphere and $2\pi/n$ on the hemisphere) and $\omega_{\min}/\omega_{\max}$ the minimum and maximum solid angle of the pixels at resolution $r_x \cdot r_y = n$. Since these ratios depend on the resolution, we give the limit value at infinity. An area preserving mapping has $A\pm[1,1]$ and a worst case mapping would have $A\pm[0,\infty]$. Where possible we give analytic derived values, otherwise we use numeric approximations.

2.3. Parametrization

The spherical coordinates for all models are parametrized after the following convention. The transformation for a normalized direction vector (x, y, z) into spherical coordinates (θ (zenith) and φ (azimuth)) and back is:



$$\begin{aligned} x &= \sin\theta\cos\varphi \\ y &= \sin\theta\sin\varphi \\ z &= \cos\theta \end{aligned} \qquad \begin{aligned} \theta &= \arccos(z) \\ \varphi &= \text{atan2}(y, x) \end{aligned}$$

2.4. Spherical Integration

At some points in this document functions are integrated over the sphere. We write

$$\int_\Omega f(\omega)\mathrm{d}\omega \qquad \text{which is equal to} \qquad \int_0^{2\pi}\int_0^{\theta_{\max}} f(\theta,\phi)\cdot\sin\theta\ \mathrm{d}\theta\ \mathrm{d}\phi.$$

$\theta_{\max}$ is either $\frac{\pi}{2}$ for an integration over the hemisphere or π for the sphere.

3. Grid Based Methods

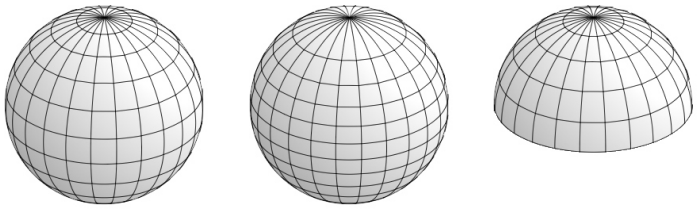
Grid based mappings transform the angular parametrization into a 2D plane and store support values at fixed positions. The function is usually reconstructed as piecewise linear interpolation of support values in the vicinity of the sampling location.

This is opposed to polynomial bases (Section 4) which define polynomials in the spherical domain and to lobe mixture models (Section 5) which have varying support locations.

3.1. Polar Coordinate Map

The polar coordinate map is parametrized over the two Euler angles θ and φ of the spherical coordinates of a direction vector.

$$u = \frac{\varphi}{2\pi} \qquad\qquad v = \frac{\theta}{\pi}$$



(a) *12x24 grid* **(b)** *Equal area* **(c)** *Hemispherical*

Figure 3. Polar Coordinate Map

Domains PCM can be used for spherical ($\theta \in [0, \pi], \varphi \in [0, 2\pi]$) and hemispherical domains ($\theta \in [0, \pi/2], \varphi \in [0, 2\pi]$) by modifying v accordingly. The resolution of θ can be halfed for hemispherical maps.

Quality The singularities at the poles introduce a strong area deviation of $\not{A}\pm[0, \pi/2]$. Pixels have different support areas leading to an oversampling in the pole region. The pixels can be made equal area by using

$$v = \frac{1}{2}\sin\left(\theta - \frac{\pi}{2}\right) + \frac{1}{2} \tag{1}$$

before computing the index. However, this results in an higher distortion (shape deviation) as visible in Figure 3.

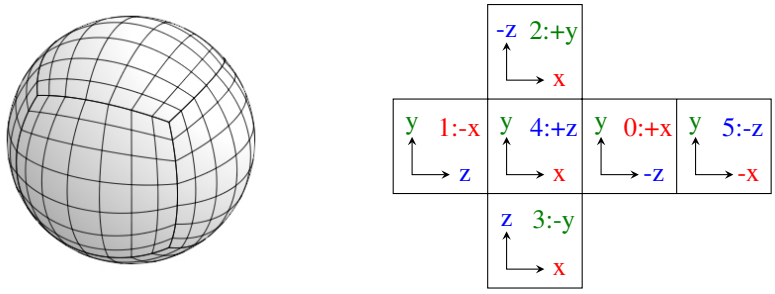
Performance Using the same angular spacing in θ and φ the resulting map resolution is $2n \times n$ and $2n \times n/2$ for spherical and hemispherical map respectively. A lookup can be performed in $\mathcal{O}(1)$ by converting the direction to spherical coordinates and computing a 2D index from the two angles. See Listing 1.

```
# get u,v coordinates in [0,1]x[0,1]
polarmap(dir) -> (u,v)
    theta = acos(dir.y)
    phi = atan2(dir.z, dir.x)
    if phi < 0: phi += 2π end
    # scale θ : [0,π] → [0,1] and φ : [0,2π) → [0,1)
    u = phi/2π
    v = theta/π
end
```

Listing 1. Polar coordinate map lookup

3.2. Cube Map

The cube map is the most common exchange format for environment maps as it is simple and provides a comparatively low distortion. It consists of 6 equally sized



(a) *6x7x7 grid* **(b)** *Face indices and coordinate mappings within faces*

Figure 4. Cube Map

squares. The mapping is done by a projection onto a surrounding cube such that the largest value of (x, y, z) becomes ± 1 and defines the face f. The two remaining coordinates define u and v as shown in Figure 4 (b).

There is an interesting extension from Tarini et al. [2004] using multiple cube maps to create low distortion texture mappings for arbitrary meshes.

Domains The cube map is mostly used for full spherical functions. It is possible to implement a hemi-cube, but the resulting difference in the face size (one square, four 2:1 rectangles) introduces an overhead.

Quality Interpolation over edges is an issue leading to discontinuities/seams on edges. One approach to solve this is to average the pixels on both sides of the edge [Isidoro and Mitchell 2005], another to warp the texture lookup function [Castaño 2012]. However, most graphics hardware supports seamless sample filters on its own. The area deviation for cube maps is $A \pm \left[2/\sqrt{3}\pi, 6/\pi \right]$.

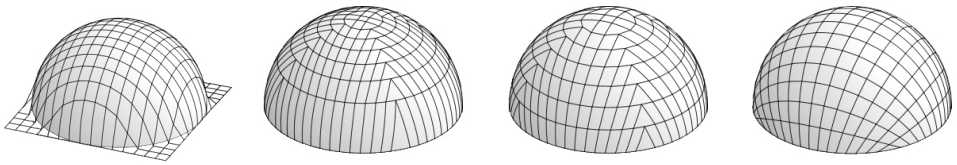
Performance The cube map takes $6 \times n \times n$ space. A lookup can be performed in $\mathcal{O}(1)$ as in Listing 2.

```
# get face index f and u,v coordinates
cubemap(dir) -> (f,u,v)
   # get 0, 1 or 2 for x, y or z
   majorDim = argmax(abs(dir))
   # map to [0,5] dependent on sign
   f = majorDim * 2 + (dir[majorDim] > 0 ? 1 : 0)
   switch majorDim:
       case 0: projU = -dir[2]/dir[0]       projV =  dir[1]/abs(dir[0])
       case 1: projU =  dir[0]/abs(dir[1])  projV = -dir[2]/dir[1]
       case 2: projU = -dir[0]/dir[2]       projV =  dir[1]/abs(dir[2])
   end
   # transform [-1,1] to [0,1]
   u = projU / 2 + 0.5
   v = projV / 2 + 0.5
```

```
end
```

Listing 2. Cube map lookup

3.3. Hemispherical Projection



(a) *16x16 grid* **(b)** *Concentric mapping Shirley [1997]* **(c)** *Concentric Lambert equal area mapping* **(d)** *Elliptical Lambert equal area mapping*

Figure 5. Hemispherical Projection

The simplest mapping from a hemisphere to a disc is the projection by removing the z component. The mapping introduces distortion towards the horizon and is wasting $1 - \pi/4 \approx 21.5\%$ of the square texture (Figure 5 (a)). The deviation of pixel sizes is $A\pm[0, \infty]$ and can be reduced by using Lambert equal area mapping for the hemisphere to the disc:

$$u = \frac{x}{\sqrt{1+z}} \qquad z = 1 - u^2 - v^2$$
$$v = \frac{y}{\sqrt{1+z}} \qquad x = u\sqrt{1+z}$$
$$y = v\sqrt{1+z}.$$

In computer graphics this mapping is also know as (dual) paraboloid mapping [Heidrich and Seidel 1998] which is the application of Lambert equal area mapping for reflection probes.

However, even with an improved projection the corners of the map are wasted which can be overcome by another disc to square mapping. Shirley and Chiu [1997] introduce the concentric map which warps the coordinates in the plane with respect to the center. The result can be seen in Figure 5 (b) for the simple projection and in Figure 5 (c) using the Lambert equal area projection. The area deviations are $A\pm\left[1/2, \infty\right]$ and $A\pm[1, 1]$ respectively. Listing 3 provides the implementation for the concentric mapping including the area correction.

In the normal compression survey by Cigolle et al. [2014], an alternative elliptic mapping is used. Semantically it pushes the corners of the square onto the disc boundary smoothly.

Disc to square:

$$t = \sqrt{(2 + u^2 - v^2)^2 - 8u^2}$$
$$u' = \frac{2u}{\sqrt{2 + u^2 - v^2 + t}}$$
$$v' = \frac{2v}{\sqrt{2 - u^2 + v^2 + t}}$$

Square to disc:

$$u = u'\sqrt{1 - \frac{v'^2}{2}}$$
$$v = v'\sqrt{1 - \frac{u'^2}{2}}$$

The mapping is shown in Figure 5 (d) in combination with the Lambert equal area projection. The elliptical disc to square mapping is not area preserving. Using Lambert equal area projection followed elliptic mapping has an deviation of $A \pm [0, 4/\pi]$.

Domains While being a hemispherical mapping by definition it can be easily extended to the sphere by using a second map if $z < 0$.

Quality Using the pure projection leads to high distortions at the horizon and to a waste of memory in the corners. Lambert equal area mapping reduces the distortion but does not solve the problem of unused corners. However, values are well distributed using the concentric or elliptical mapping.

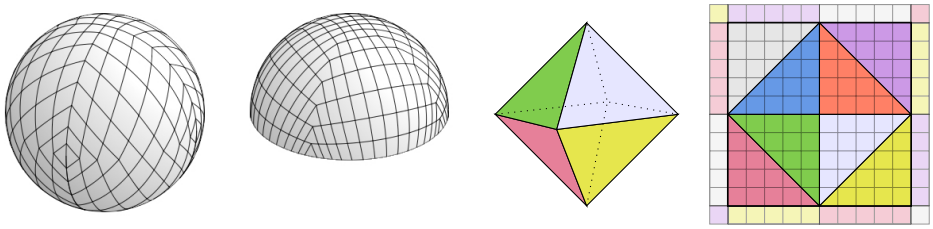
Performance The map requires a single $n \times n$ texture for hemispherical functions and twice that many for a sphere. A lookup can be performed in $\mathcal{O}(1)$ regardless of the chosen mapping combination.

```
# Map a hemisphere direction to u,v coordinates.
hemiproj(dir) -> (u,v)
    phi = atan2(dir.y, dir.x)
    if phi < -π/4: phi += 2π end
    # Make concentric rings equally spaced.
    # No need to divide by sqrt(1+dir.z)!  Simpler in polar coordinates:
    r = sqrt(1 - dir.z)
    # Transformation from Shirley and Chiu 1997
    if phi < π/4:     a =  r   b = phi * r / π * 4
    elif phi < 3π/4: b =  r   a = (π/2 - phi) * r / π * 4
    elif phi < 5π/4: a = -r   b = (π - phi) * r / π * 4
    else             b = -r   a = (phi - 3π/2) * r / π * 4
    end
    u = a / 2 + 0.5
    v = b / 2 + 0.5
end
```

Listing 3. Concentric equal area hemisphere map lookup

3.4. Octahedral Maps

Octahedral maps have the simplest mapping of Platonic solids. They were used for environment maps [Engelhardt and Dachsbacher 2008] and normal vector encoding [Cigolle et al. 2014] before.



(a) *16x16 grid* **(b)** *Hemispherical 16x16 grid* **(c)** *Mapping from the octahedron to a square including repeated pixels for interpolation*

Figure 6. Octahedral Map Projection

Similar to cube maps directions are projected using Manhatten norm L_1 instead of infinity norm L_∞. Then the upper hemisphere is projected to the plane and the lower one can be unfolded to fill the remaining corners of a square as shown on the right of Figure 6.

Domains The octahedral map can be used for spheres and hemispheres with slightly different implementations. For hemispheres the projection must be rotated by $45°$ to fill the square (see Listing 4).

Quality Octahedral maps have a distortion similar to the concentric hemispherical projection but lack their equal area sized pixel. The area deviation is $A \pm \left[1/\pi, 3\sqrt{3}/\pi\right]$ in both domains. Seamless interpolation is possible by repeating pixels along borders by using $180°$ rotations of the same map.

Performance The map requires a single $n \times n$ texture for both domains. A lookup can be performed in $\mathcal{O}(1)$. Without a hardware support, which is usually given for cube maps, this is among the fastest mappings.

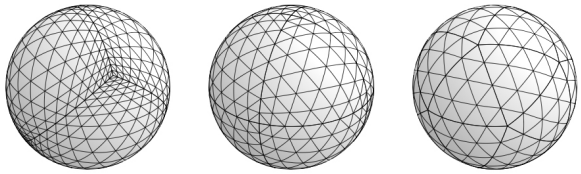
```
# map a direction from the sphere to u,v
octamap(dir) -> (u,v)
    l1norm = abs(dir.x) + abs(dir.y) + abs(dir.z)
    if dir.z >= 0:
        u = dir.x / l1norm
        v = dir.y / l1norm
    else # warp lower hemisphere
        u = (1 - dir.y / l1norm) * (dir.x >= 0 ? 1 : -1)
        v = (1 - dir.x / l1norm) * (dir.y >= 0 ? 1 : -1)
    end
    u = u / 2 + 0.5
    v = v / 2 + 0.5
end

# map a direction from the upper hemisphere to u,v
hemioctamap(dir) -> (u,v)
    l1norm = (abs(dir.x) + abs(dir.y) + abs(dir.z)) * 2
    u = (dir.x - dir.y) / l1norm + 0.5
```

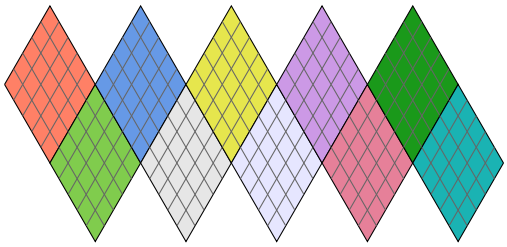
```
    v = (dir.x + dir.y) / l1norm + 0.5
end
```

Listing 4. Octahedral map lookup

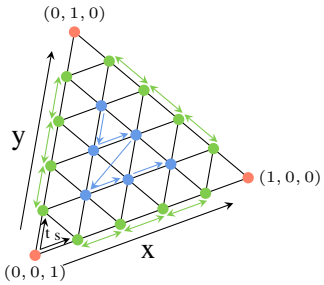
3.5. Tetrahedral and Icosahedral Maps



(a) *Tesselated tetrahedron (n=16), octahedron (n=8) and icosahedron (n=4), where n is the number of subdivisions.*



(b) *Alternative mapping of the icosahedron onto 10 squares. This enables hardware interpolation, but suffers from many seams.*



(c) *Indexing within a triangle to map triangle vertices to memory without seams. Barycentric coordinates α and β are remapped to indices (x,y) and local barycentric coordinates (s,t). Shared data on vertices (red) and edges (green) is found via lookup table. The order of interior vertices (blue) is* $(n - y - 2) \cdot (n - y - 1) + x - 1$.

Figure 7. Triangulated sphere mapping

The tetrahedron and the icosahedron are two more Platonic solids which subdivide the sphere into regular faces. Unlike cube and octahedral maps, coordinates cannot be projected to a square or rectangle easily. While the tetrahedron net could be unfolded into a rectangle this is not possible for the icosahedron. Figure 7 (b) shows that the icosahedron can be mapped to 10 squares, which is similar to the cube map and octahedral map implementations. While hardware interpolation is possible the number of seams which require a duplication of data increases.

Therefore, we implemented a triangle mapping by first detecting the face and then mapping barycentric coordinates onto linear indices. Interpolation within a triangle is possible using the barycentric coordinates. To interpolate over boundaries our implementation stores data on the vertices of the grid (colored dots in Figure 7 (c)). To avoid redundancy, knowledge of the neighborhood is incorporated via lookup table (LUT). The LUTs have a size proportional to the number of primary triangles (4 and 20) and do not depend on tesselation. Vertices on edges need to be iterated in the same

order in both adjacent triangles. To solve this we introduced a third LUT `EDGE_DIR` which simply assigns each edge a unique direction. For details see Listing 6.

Domains Tetrahedral and icosahedral maps can only be used for the spherical domain. However, our triangle mapping allows more arbitrary shapes, but since the size of the LUTs depends on the number of faces it is not reasonable for a more complex scenario.

Quality The increasing number of faces reduces distortion and area deviation compared to other similar mappings. The icosahedral map has an area deviation of $A\pm\left[1/2, 1.207\right]$ There is no gain in using tetrahedral maps ($A\pm[0.061, 3.308]$).

Performance The storage cost for the triangle mapping is $f * (n^2 - 1)/2 + v$, where f is the number of faces, v the number of vertices and n the tessellation. On top of that the LUTs require $\mathcal{O}(f)$ space with a small factor. A lookup can be performed in $\mathcal{O}(f)$, because the face must be found first.

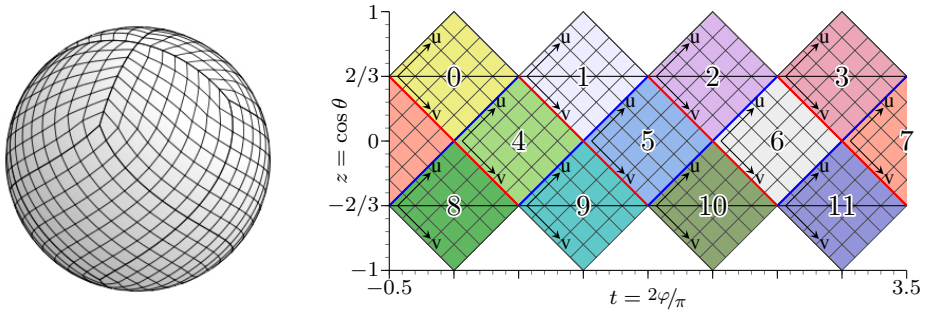
```
# Map an integer grid coordinate x,y to linear memory
# and handle redundant vertices between triangles.
# Thereby n is the number of subdivisions per triangle.
adress(t, x, y) -> i
    # Map boundary edges explicitly with LUTs
    if x==0 && y==0: i = VERTEX_OFFSETS[t*3+0]
    elif x==n:       i = VERTEX_OFFSETS[t*3+1]
    elif y==n:       i = VERTEX_OFFSETS[t*3+2]
    elif x==0:       i = EDGE_OFFSETS[t*3+0] + EDGE_DIR[t*3+0] ? y-1 : n-y-1
    elif y==0:       i = EDGE_OFFSETS[t*3+1] + EDGE_DIR[t*3+1] ? x-1 : n-x-1
    elif x+y==n:     i = EDGE_OFFSETS[t*3+2] + EDGE_DIR[t*3+2] ? x-1 : y-1
    else # interior vertex inside triangle
        x = x - 1
        y = n - y - 2
        i = t * (n-2) * (n-1) / 2 + y * (y+1) / 2 + x
    end
end

# get a linear interpolated sample from triangulated sphere
trianglemap(dir) -> sample
    for t in triangles:
        if _,α,β,γ = intersects(dir, t):
            x,s = intfrac(α * n)
            y,t = intfrac(β * n)
            if s > t && x < y: # upper right triangle
                return data[adress(t, x+1, y+1)] * (s + t - 1)
                     + data[adress(t, x+1, y  )] * (1 - t)
                     + data[adress(t, x,   y+1)] * (1 - s)
            else
                return data[adress(t, x,   y  )] * (1 - s - t)
                     + data[adress(t, x+1, y  )] * s
                     + data[adress(t, x,   y+1)] * t
            end
        end
    end
```

```
end
```

Listing 5. Interpolated lookup on triangulated spheres

3.6. HEALPix



(a) *8x8x12 grid. All pixels span the same solid angle.*

(b) *The sphere is partitioned into equatorial and polar zones at* $\cos\theta = \pm 2/3$ *and within the equatorial zone by the curves described by Eq. 2 and 3.*

Figure 8. HEALPix mapping

The HEALPix (Hierarchical Equal Area isoLatitude Pixelation) mapping originates in astronomic applications by the NASA [Gorski et al. 2005]. Its 12 curvilinear quadrilaterals can be tessellated perfectly into equal area pixels. Because of its low discrepancy of pixel locations it was also used for environment mapping before [Wan et al. 2005].

First, the sphere is divided into an equatorial and two polar zones at $\cos\theta = \pm 2/3$, denoted by the two black horizontal lines in Figure 8 (b). Within the equatorial zone, the sphere is partitioned by

$$k = -\frac{1}{2} + t + \frac{3}{4}\cos\theta \qquad k = 0, 1, 2, 3 \tag{2}$$

$$\text{and} \qquad l = -\frac{1}{2} + t - \frac{3}{4}\cos\theta \qquad l = 0, 1, 2, 3 \tag{3}$$

where $t = 2\varphi/\pi$ and the curves of Eq. 2 and 3 are shown in red and blue respectively. The fractional parts between the curves map directly onto u and v.

The polar zones are divided into four triangular faces by

$$k = t \qquad\qquad k = 0, 1, 2, 3.$$

The u and v coordinates must be aligned with the parts in the equatorial zone, which yields

$$u = \sqrt{3(1 - |\cos\theta|)} \cdot t$$
$$v = 1 - \sqrt{3(1 - |\cos\theta|)} \cdot (1 - t),$$

where u and v are swapped for the northern cap.

Domains HEALPix is designed for the spherical domain. Dividing it into two hemispheres would lead to a mixture of triangle and quad faces.

Quality The map has equal area pixels ($A \pm [1, 1]$) regardless of the subdivision at a comparable low distortion of pixels. Its quadrilateral faces are well suited for quad-tree subdivisions. Interpolation has the issue of seams between faces. This can be solved by repeated pixels or a lookup based neighborhood assignment, again.

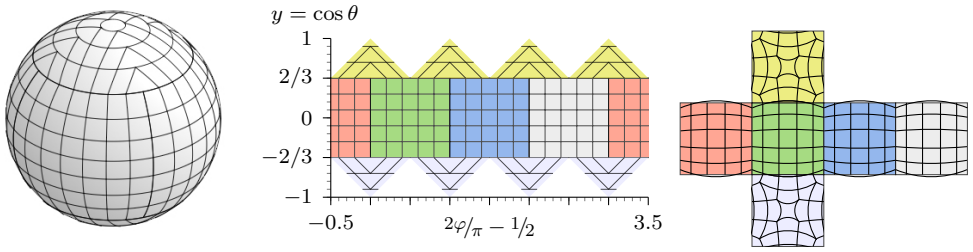
Performance HEALPix requires $f \times n \times n$ space and can be looked up in $\mathcal{O}(1)$.

```
# Get the face f and the u, v coordinates matching Figure 8 (b)
healpix(dir) -> (f,u,v)
    t = atan2(dir.y, dir.x) * 2 / pi
    # equatorial zone?
    if abs(dir.z) <= 2.0/3.0:
        k,u = intfrac(mod(3.5 + t + 0.75 * dir.z, 4.0))
        l,v = intfrac(mod(3.5 + t - 0.75 * dir.z, 4.0))
        if k == l:            f = 4 + k  # = 4 + l -> central quad
        elif k == mod(l+3,4): f = 8 + l  # southern triangle
        else                  f = k      # l==mod(k+3,4) -> northern triangle
    else
        f,x = intfrac(t)
        x = sqrt(3 * (1 - abs(dir.z)))
        if dir.y < 0:
            f += 8
            u = x * t
            v = 1 - x * (1 - t)
        else
            u = 1 - x * (1 - t)
            v = x * t
        end
    end
end
```

Listing 6. HEALPix lookup

3.7. Isocube

The Isocube is a map inspired by HEALPix [Wan et al. 2007]. It extends the idea such that hardware interpolation can be used. The trick is a remapping of a direction such that a lookup in a usual cube map (Section 3.2) gives equal area samples on the sphere. Figure 9 (b) shows a very similar parametrization as the HEALPix mapping (Figure 8 (b)). Again, the sphere is divided into an equatorial zone between $\cos\theta = \pm 2/3$ and two polar regions. The mapping within the polar zones is very similar to the concentric mapping of Shirley and Chiu [1997] (Section 3.3).



(a) *6x7x7 grid* **(b)** *Unfolding in parameter space.* **(c)** *Distortion of the cube map through the remapping.*

Figure 9. Isocube Map

The equatorial zone is partitioned by the curves

$$u = \left(\frac{2}{\pi}\varphi + \frac{1}{2}\right) \mod 1$$
$$\text{and} \qquad v = \frac{3}{4}y + \frac{1}{2}.$$

Within each of the four triangular faces divided by $\frac{k}{2\pi}\varphi, k = 0, 1, 2, 3$ the curves are

$$a = \sqrt{3 - 3|\cos\theta|}$$
$$\text{and} \qquad b = \frac{2}{\pi}\varphi \cdot a.$$

Dependent on k the coordinate $(a, b)^T$ must be transformed into $(u, v)^T$ with one of the following matrices A_k.

$$A_0 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \quad A_1 = \begin{bmatrix} 2 & -2 \\ 1 & 0 \end{bmatrix} \quad A_2 = \begin{bmatrix} -1 & 0 \\ 4 & -2 \end{bmatrix} \quad A_3 = \begin{bmatrix} -6 & 2 \\ -1 & 0 \end{bmatrix}$$

Using u, v and z a point on the unit cube can easily be constructed and used as lookup direction in a cube map. This is demonstrated in Listing 7.

Domains The Isocube is a pure spherical map.

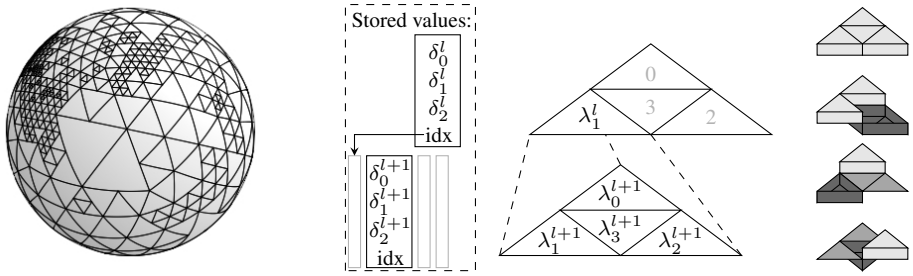
Quality This map has equal area pixels ($\mathcal{A}\pm[1, 1]$) like HEALPix, but adds distortion in the polar regions.

Performance The $\mathcal{O}(1)$ lookup time is similar to that of HEALPix, but offers the opportunity of hardware interpolation. The space requirement is the same as for cube maps (Section 3.2).

```
# Remap a direction to a lookup coordinate in a cube map.
# The intermediate results u and v can be used for manual adressing, if
# an additional face f is determined.
isoDirToCubeDir(dir) -> cubeDir
    # Compute azimuth angle and convert to [0.5,3.5).
    t = 2 / π * atan2(dir.y, dir.x)
    if t < -0.5: t += 4
    # Equatorial zone?
    if abs(dir.z) <= 2.0/3.0:
        a = 1
        b = t
    else
        a = sqrt(3 * (1 - abs(dir.z)))
        b = t * a
    end
    # Transform:  kind of a rotation around up axis
    k = floor(t + 0.5)
    [u,v] = A[k] * [a,b]
    # Produce cube coordinate
    cubeDir = [u, v, clamp(dir.z * 1.5, -1, 1)]
end
```

Listing 7. Isocube direction remapping

3.8. Spherical Wavelets



(a) *Icosahedron based wavelet subdivision.*

(b) *Haar-Wavelet nomenclature for one subdivision from level l to $l+1$. λ values are (averaged) function values, δ are coefficients of the basis functions. Right: basis functions of H (Eq. 4).*

Figure 10. Spherical wavelet subdivision

A wavelet is a basis function with local support in both space and frequency domain [Grossmann and Morlet 1984]. Wavelets and scaling functions together represent a function f at different levels of detail by scaling and translating the basis (multiresolution analysis). The wavelet transformation (analysis) is the decomposition of f into a high frequency part and a low frequency part recursively. The resulting difference between high-pass and low-pass filtered version is stored into δ coefficients. Hence, many coefficients are close to zero and may be discarded for lossy compres-

sion purposes. The reconstruction (synthesis) of the function is done by adding the differences and low-frequency parts recursively.

To apply the wavelet transformation on a sphere a subdivision scheme with access to the local neighborhood is required. The easiest way is to use a sphere to rectangle mapping (Polar coordinates Sec. 3.1, Projections Sec. 3.3 or Octahedral mapping Sec. 3.4) and perform a standard transformation in the plane. Lalonde [1997] used this approach for the compression of BRDFs.

Lounsbery et al. [1997] introduced the wavelet transformation on arbitrary meshes to compress the meshes themselves. Schröder and Sweldens [1995] used a similar approach based on triangular subdivisions of the sphere to compress spherical functions. They provides a mathematically framework on abstract index sets and introduce the lifting method from Sweldens [1998] on the sphere. They tested different setups and came to the conclusion that storing the data on vertices, not on faces, performs best. We implemented what they call the lineal lifted wavelet basis and the Haar-Wavelet basis. Both are based a triangular quad-tree on the faces of an icosahedron.

The Haar-Wavelet is the simplest Wavelet in form of a square-function on the unit interval. No neighborhood is required for analysis and synthesis. In our implementation each node in the coefficient (quad-) tree stores three δ values and a single child index. Thus the overhead of tree indices is small compared to the data. On the topmost level a single function value λ and a child pointer is stored per face. The four values in a tessellated triangle are transformed using the Haar-transformation matrix:

$$H = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix}$$

$$[\lambda^l, \delta_0^{l+1}, \delta_1^{l+1}, \delta_2^{l+1}]^T = H \cdot [\lambda_0^{l+1}, \lambda_1^{l+1}, \lambda_2^{l+1}, \lambda_3^{l+1}]^T. \tag{4}$$

The inverse transformation during reconstruction is H^T, because H forms an orthonormal basis. Figure 10 (b) visualizes the quad tree and its indexing scheme. Further, the basis functions spanned by H are shown.

The lineal basis is implemented similarly. Function values λ and child indices are stored on the twelve vertices of the icosahedron. Each node stores three δ values, one on each edge, and a child index. Figure 11 shows the index names used in the following formulas. The double indexed m values are only needed if lifting is applied.

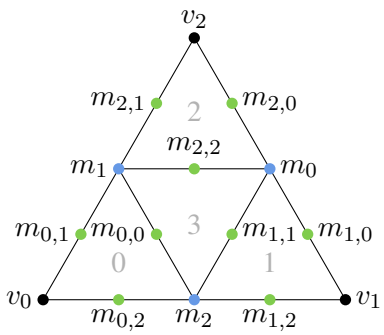


Figure 11. Index set of vertex basis

The linear basis analysis is

$$\lambda^l_{v_i} = \lambda^{l+1}_{v_i} \tag{5}$$

$$\delta^{l+1}_{m_i} = \lambda^{l+1}_{m_i} - \frac{1}{2}(\lambda^{l+1}_{v_j} + \lambda^{l+1}_{v_k}) \tag{6}$$

$$\text{for } i,j,k = 0,1,2 \wedge i \neq j \neq k$$

which could also be expressed by a sparse 6×6 matrix. Note that this is the transformation without lifting.

To improve the compression abilities the Wavelet base can be lifted [Schröder and Sweldens 1995]. This is introduced by an additional transformation of λ_m values before applying Equation 6.

$$\tilde{\lambda}^{l+1}_{m_0} = \lambda^{l+1}_{m_0} + \frac{1}{4}\left(\delta^{l+2}_{m_{2,0}} + \delta^{l+2}_{m_{2,2}} + \delta^{l+2}_{m_{1,1}} + \delta^{l+2}_{m_{1,0}}\right) \tag{7a}$$

$$\tilde{\lambda}^{l+1}_{m_1} = \lambda^{l+1}_{m_1} + \frac{1}{4}\left(\delta^{l+2}_{m_{0,1}} + \delta^{l+2}_{m_{0,0}} + \delta^{l+2}_{m_{2,2}} + \delta^{l+2}_{m_{2,1}}\right) \tag{7b}$$

$$\tilde{\lambda}^{l+1}_{m_2} = \lambda^{l+1}_{m_2} + \frac{1}{4}\left(\delta^{l+2}_{m_{1,2}} + \delta^{l+2}_{m_{1,1}} + \delta^{l+2}_{m_{0,0}} + \delta^{l+2}_{m_{0,2}}\right) \tag{7c}$$

Domains The domain can be chosen arbitrary due to a choice of primary triangles. Alternative to the triangle based implementation, a standard wavelet transformation is possible on the quad faces of a cube map or HEALPix.

Quality Both implementations have discontinuity artifacts among different tessellated neighbored triangles (T-junctions). It would be possible to track neighbors in the traversal to solve this problem.

Performance In an adaptive implementation the overhead of tree pointers/indices wastes some memory. Additionally, our vertex base implementation stores δ values on m_i locations twice if triangles on both sides of the edge are subdivided. Alternatively, the transformation could be done on a regular tessellated sphere prior to some entropy encoding to solve both problems. However, encoding the densely transformed function requires a decoding step before sampling. A lookup can be performed in $\mathcal{O}(f+l)$ where f is the number of primary faces and l the maximum subdivision level.

```
# Get the index of the child triangle c and the barycentric coordinates of
# the sampling point within this triangle
traverse(α, β, γ) -> (c, αc, βc, γc)
    x,αc = intfrac(predecessor(2) * α) # using predecessor(2) ensures x={0,1}
    y,βc = intfrac(predecessor(2) * β)
    z,γc = intfrac(predecessor(2) * γ)
```

```
    c = (1-x) * (1 + (1-y) * (1 + (1-z)))
    if c == 3: # center triangle -> invert barycentrics
        αc,βc,γc = 1-αc, 1-βc, αc+βc-1
end

haar(dir) -> sample
    for t in triangles: # Find the primary triangle.
        if _,α,β,γ = intersects(dir, t):
            node, sample = t.node, t.λ
            while valid(node.idx)
                c,α,β,γ = traverse(α,β,γ)
                node = nodes[node.idx + c]
                # take column c of H for inverse transformation
                sample = [sample, node.δ0, node.δ1, node.δ2] * H[][c]
            end
            return
end end end

linearwavelet(dir) -> sample
    for t in triangles: # Find the primary triangle.
        if _,α,β,γ = intersects(dir, t):
            # load top level data and start recursion
            λ0, λ1, λ2 = λ[t.v0], λ[t.v1], λ[t.v2]
            node = t.node
            while node
                c,α,β,γ = traverse(α,β,γ)
                # Reconstruct the three values on the edges.
                λm0 = node.δ0 + (λ1 + λ2) / 2
                λm1 = node.δ1 + (λ0 + λ2) / 2
                λm2 = node.δ2 + (λ0 + λ1) / 2
                # Optionally invert lifting.
                if valid(node.idx]
                    c0,c1,c2 = nodes[node.idx+0], ..., nodes[node.idx+2]
                    λm0 -= (c2.δ0 + c2.δ2 + c1.δ1 + c1.δ0) / 4
                    λm1 -= (c0.δ1 + c0.δ0 + c2.δ2 + c2.δ1) / 4
                    λm2 -= (c1.δ2 + c1.δ1 + c0.δ0 + c0.δ2) / 4
                end
                # Select the λ for proceeding with the child.
                switch(c):
                    0: λ0, λ1, λ2 =  λ0, λm2, λm1
                    1: λ0, λ1, λ2 = λm2,  λ1, λm0
                    2: λ0, λ1, λ2 = λm1, λm0,  λ2
                    3: λ0, λ1, λ2 = λm0, λm1, λm2
                end
                node = nodes[node.idx + c] # invalid index -> null
            end
            return α * λ0 + β * λ1 + γ * λ2
end end end
```

Listing 8. Sampling of adaptive Wavelet quad-tree (synthesis)

4. Polynomial Bases

Polynomial bases are projections onto orthogonal basis functions of different frequency. The original function is expressed as a linear combination of basis functions

b with scalar coefficients c:

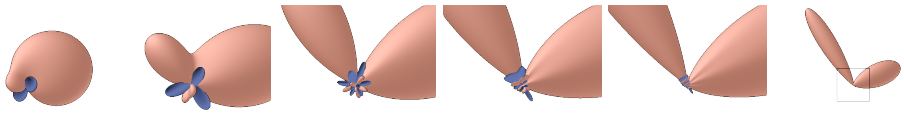
$$f(x) = \sum_i c_i \cdot b_i(x). \tag{8}$$

A function is reconstructible if the frequency of basis b is twice as high as the highest frequency in f (Nyquist-Shannon sampling theorem [Nyquist 1928; Shannon 1948]). The orthogonality of bases ($\int b_i \cdot b_j = 0$) allows a fast projection of f into the basis:

$$c_i(x) = \int_{\Omega_i} b_i(\omega) \cdot f(\omega) \mathrm{d}\omega, \tag{9}$$

where Ω_i is the domain in which the basis function b_i is unequal to zero. All models in this section have bases with global support. Therefore, the domain Ω_i is always the hemisphere 2πsr or sphere 4πsr.

4.1. Artifacts: Ringing and Smoothing



(a) *3 bands* **(b)** *4 bands* **(c)** *6 bands* **(d)** *8 bands* **(e)** *12 bands* **(f)** *input f*

Figure 12. Spherical harmonic fit of a function f with two peaks. The integral over all representations is the same, but images (a)-(e) are scaled by a factor of 5. With more bands, the ringing is decreased in magnitude and increased in frequency. Further, fewer bands cause higher smoothing up to the loss of the two peaks as visible in (a).

All polynomial bases suffer from the same artifact, namely ringing. The removal of high-frequency bands causes an overshooting of function values in lower bands. I.e. high frequency peaks in f cause high coefficients for some bases which results in a repetition of the feature over the full sphere with a pattern of the base. An unlimited number of bands compensates the overshooting with the higher frequency bases.

A common approach to solve this problem is to blur the function in form of windowing in the frequency domain [Sloan 2008]. Therefore, a weight w is applied to all coefficients based on the band index l they live in. We use Lanczos windowing in some of the following examples and evaluations:

$$w_{\text{Lanczos}}(l) = \sin\left(\frac{\pi l}{n}\right) \cdot \frac{n}{\pi l}. \tag{10}$$

The first band has $w_{\text{Lanczos}}(0) = 1$ and the last used band $l = n - 1$ has a value slightly greater zero.

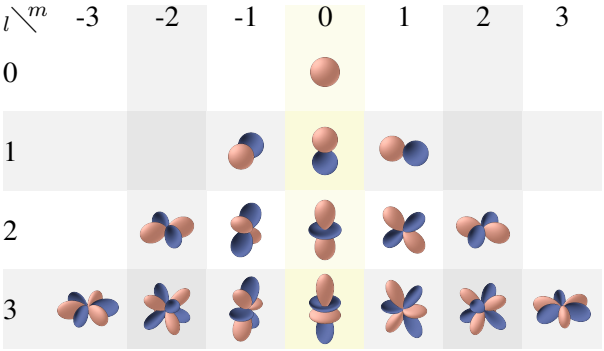


Figure 13. Polynomials of the first four bands of the SH basis. Red and blue shades denote the sign. The central column shows the zonal harmonics y_l^0.

4.2. Spherical Harmonics

Spherical harmonics is the most used polynomial basis in computer graphics. It is used for Precomputed Radiance Transfer [Sloan et al. 2002], sky lights and caches for local light [Krivanek et al. 2005; Vardis et al. 2014; Jendersie et al. 2016]. Green [2003] and Sloan [2008] both give a practical introduction to SH in computer graphics. The success of the basis is motivated by the following properties:

- The SH basis is rotational invariant. I.e. rotating f before projection yields the same as rotating the SH functions themselves. This is not true for any of the grid based models.
- The projection of one SH onto another is the dot product of their coefficient vectors: $\sum a_i \cdot b_i$. This makes SH evaluations very fast for a few bands.
- There is no need for a fitting or sampling process. A function can simply be projected (see below).

The original definition of basis functions is based on complex numbers similar to the Fourier Transformation and goes back to P.S. Laplace 1782. In computer graphics only the real-valued fraction is used and bases are denoted with y_l^m. Thereby, $l \in [0, \infty)$ is the index of the band, with greater l representing higher frequencies, and $m \in [-l, l]$ is an index within the band. Hence, a band has $2l + 1$ basis functions leading to a total of n^2 coefficients for an n-band SH. The bases with $m = 0$ are also called zonal harmonics. These are isotopic with respect to the up axis as visible in Figure 13. The basis polynomials are defined by

$$y_l^m(\theta, \varphi) = \begin{cases} \sqrt{2} K_l^{|m|} \cdot \cos |m| \varphi \cdot P_l^{|m|}(\cos \theta) & m > 0 \\ \sqrt{2} K_l^{|m|} \cdot \sin |m| \varphi \cdot P_l^{|m|}(\cos \theta) & m < 0 \\ K_l^0 \cdot P_l^0(\cos \theta) & m = 0 \end{cases} \tag{11}$$

with K being a normalization factor and P being the associated Legendre polynomials:

$$K_l^m = \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}}$$

$$P_m^m(x) = (-1)^m(1-x^2)^{m/2}\prod_1^m(2m-1) \tag{12a}$$

$$P_{m+1}^m(x) = x(2m+1)P_m^m \tag{12b}$$

$$P_l^m(x) = x\left(\frac{2l-1}{l-m}\right)P_{l-1}^m - \left(\frac{l+m-1}{l-m}\right)P_{l-2}^m. \tag{12c}$$

Equation 12c is used as long as $l > m+1$, then recursion is finished by 12a and 12b. Thus, each function is defined by $l-m+1$ steps, always going up in the pyramid until reaching the spine.

Domains SH are defined for spherical domain only. When used for hemispherical functions the second hemisphere must be filled with reasonable values. Using negative values $f(\pi-\theta,\phi) = -f(\theta,\phi)$ forces half the coefficients to be zero. All functions with even $m+l$ are symmetric to the x,z-plane and are canceled by the choice above

Quality See Section 4.1 to ringing.

Performance Due to global support a lookup takes $\mathcal{O}(n^2)$ steps with n being the number of bands.

The code shown in Listing 10 is a general optimized evaluation approach. In practice it is advisable to use the code generator from Sloan [2013] which unrolls the recursion and summarizes as much as possible.

```
evalsh(dir) -> s
    s = 0
    sinTheta = sqrt((1 - dir.z) * (1 + dir.z)) # sqrt(1-cos²θ)
    sinThetaM = 1  # sin^m θ for m=0
    # Use addition theoreme for cos|m|φ and sin|m|φ.
    cosPhi = dir.x / sinTheta    cosMPhi = cosPhi
    sinPhi = dir.y / sinTheta    sinMPhi = sinPhi
    for m in [0,n-1] # n == number of bands
        p_m_m = FACTORIAL[m] * sinThetaM   # Eq.(12a)
        i = m * (m + 1)
        # For zonal harmonics m==0 do not evaluate twice:
        if m != 0: s += c[i + m] * p_m_m * cosMPhi * K[i/2 + m]
                   s += c[i - m] * p_m_m * sinMPhi * K[i/2 + m]
        if m < n-1:
            p_m1_m = dir.z * (2 * m + 1) * p_m_m   # Eq.(12b)
            i = (m + 1) * (m + 2)
            if m != 0: s += c[i + m] * p_m1_m * cosMPhi * K[i/2 + m]
                       s += c[i - m] * p_m1_m * sinMPhi * K[i/2 + m]
```

```
            for l in [m+2, n-1]:
                p_l_m = (dir.z*(2*l-1)*p_m1_m-(l+m-1)*p_m_m)/(l-m) # Eq.(12c)
                i = l * (l + 1)
                if m != 0: s += c[i + m] * p_l_m * cosMPhi * K[i/2 + m]
                           s += c[i - m] * p_l_m * sinMPhi * K[i/2 + m]
                p_m_m, p_m1_m = p_m1_m, p_l_m
            end
        end
        # Evaluate next power (-1√(1-x²))^m for p_m_m
        sinThetaM *= -sinTheta
        # Update angles with addition theorme
        nextSinMPhi = sinPhi * cosMPhi + cosPhi * sinMPhi
        nextCosMPhi = cosPhi * cosMPhi - sinPhi * sinMPhi
        sinMPhi, cosMPhi = nextSinMPhi, nextCosMPhi
    end
end
```

Listing 9. SH evaluation. The same code can be used for projection with the use of c[i±m] += ... instead of s += c[i±m] ... `K` is an array of precomputed normalization constants and `FACTORIAL` an array of the product terms from Equation 12a.

4.3. Hemispherical Harmonics

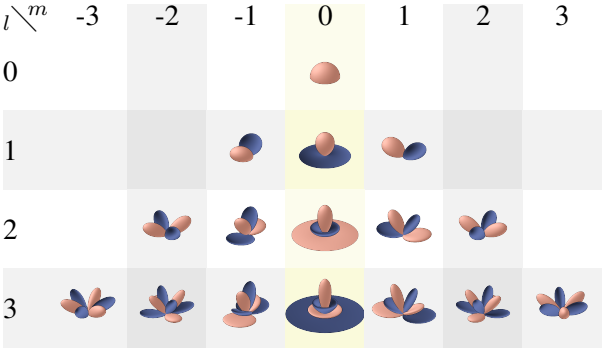


Figure 14. HSH basis functions for the first four bands. Red and blue shades denote the sign.

Hemispherical harmonics are shifted versions of the SH basis. They were defined by Gautron et al. [2004] with a linear transformation of $\cos\theta$ in the SH basis. Using $x \to 2x-1$ for the input argument of the associated Legendre Polynomials (Equation 12) shifts the south pole to the equator. The result is a new orthogonal base in the positive hemisphere. All basis polynomials, except the former zonal harmonics, are zero for $\theta = \pi/2$. Therefore, the HSH basis can only represent a constant value on the entire equator. An alternative is the $\mathcal{H}$ basis (Section 4.4).

Together with a shift of P_l^m the normalization constant K_l^m must be changed.

Replacing these two terms in the SH definition (Equation 11) with

$$\tilde{P}_l^m(x) = P_l^m(2x-1)$$
$$\tilde{K}_l^m = \sqrt{\frac{2l+1}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}} \tag{13}$$

gives the basis polynomials $h_l^m(\theta, \varphi)$ of HSH basis.

Quality Additional to ringing (Section 4.1) the HSH introduces a strong radial blur towards grazing angles. In the limit (equator itself) there is only a constant value independent of φ.

Performance Projection and lookup are the same as for SH: $\mathcal{O}(n^2)$.

The implementation follows that of SH (Listing 10). The only necessary changes are the use of $dir.z * 2 - 1$ instead of $dir.z$ in all locations.

4.4. $\mathcal{H}$ Basis (Generalized)

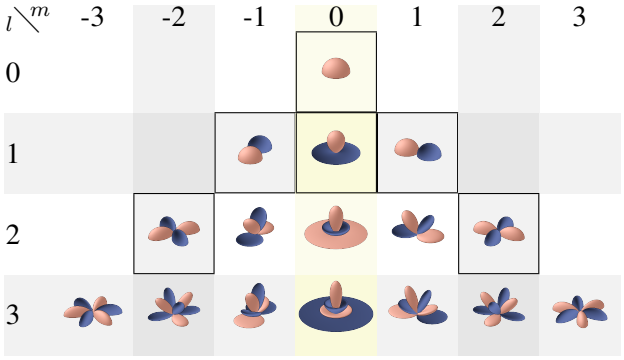


Figure 15. Polynomials of the first four bands of the $\mathcal{H}$ basis. Red and blue shades denote the sign. The framed bases are from Habel and Wimmer [2010], whereas all others result from our generalized form.

The $\mathcal{H}$ basis from Habel and Wimmer [2010] is a mixture between SH and HSH. Habel et al. observed, that the SH basis wastes some information if used for the hemispherical domain and that HSH cannot represent different values on the boundary. Instead they used the first shifted zonal harmonic $\tilde{y}_1^0$ from HSH basis and the xy-plane symmetric $y_m^{\pm m}$ from SH basis up to the second band. All other basis functions were rejected. Therefore, the original $\mathcal{H}$ basis is only defined for four ($\mathcal{H}_4$ without $y_2^{\pm 2}$) and for six coefficients ($\mathcal{H}_6$).

We generalized this idea by shifting all basis functions of the SH basis individually. The shift s_l^m can be chosen such that the peak of the most z-negative lobe is lifted to the equator. I.e. now, all lobes are at least partially in the upper hemisphere

and the lowest lobes intersect the xz-plane in their zenith. For 2 bands this gives $\mathcal{H}_4$. The former $\mathcal{H}_6$ is a subset of $\mathcal{H}_9$ with 3 bands.

The basis polynomials $H_l^m(\theta, \varphi)$ are defined as the SH definition (Equation 11) using

$$s_l^m = \sqrt{\frac{l - |m|}{l}} \tag{14}$$
$$\hat{P}_l^m(x) = P_l^m((1 + s) \cdot x - s)$$

instead of P_l^m and using the normalization factor for hemispheres $\tilde{K}_l^m$ from HSH basis (Equation 13).

The $\mathcal{H}$ basis is orthogonal for $\mathcal{H}_4$ and $\mathcal{H}_6$, but looses the orthogonality otherwise. Functions within a band l remain orthogonal, due to $\sin$ and $\cos$ being orthogonal, but bases which share the index $m \neq 0$ are not (columns in Figure 15). The Zernike base (Section 4.5) is similar, but retains the orthogonality.

Quality Additional to ringing (Section 4.1) the generalized $\mathcal{H}$ basis is not orthogonal making it more difficult to fit.

Performance The lookup is the same as for SH: $\mathcal{O}(n^2)$.

4.5. Zernike's Basis

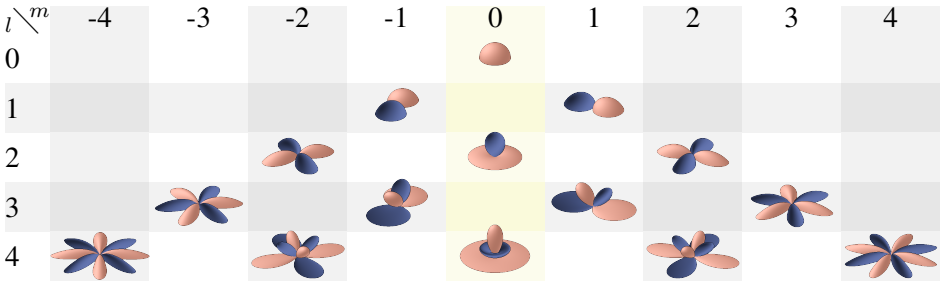


Figure 16. Polynomials of the first five bands of the Zernike basis. The band structure is different from that of the other bases as there are only $l + 1$ basis functions per band. Red and blue shades denote the sign.

This basis was introduced by Zernike [1934] in the context of mirror optics analysis. It uses orthogonal polynomials which are defined on a disc with coordinates (ϕ, r), closely related to Jacobi polynomials. It can be extended to the hemisphere with

$$r(\theta) = \sqrt{2} \sin \frac{\theta}{2} \tag{15}$$

as done by Koenderink et al. [1996] to represent BRDFs. The resulting basis is again an orthonormal system as shown by Koenderink.

The definition looks very similar to that of the SH basis:

$$Z_l^m(\theta,\varphi) = \begin{cases} \sqrt{\frac{l+1}{\pi}}\cos(|m|\varphi)\cdot R_l^{|m|}(r(\theta)) & m>0 \\ \sqrt{\frac{l+1}{\pi}}\sin(|m|\varphi)\cdot R_l^{|m|}(r(\theta)) & m<0 \\ \sqrt{\frac{l+1}{2\pi}}R_l^0(r(\theta)) & m=0 \end{cases}$$

$$R_l^m(x) = \sum_{k=0}^{(l-m)/2} \frac{(-1)^k(l-k)!}{k!\left(\frac{l+m}{2}-k\right)!\left(\frac{l-m}{2}-k\right)!}\cdot x^{l-2k} \tag{16}$$

with two differences. First, it uses different polynomials R_l^m instead of Legendre polynomials. The second difference is another band structure, which is best visible in Figure 16. Each band has $l+1$ basis polynomials. Therefore, there are $n(n+1)/2$ basis functions in total, allowing a finer grained control of the number of required coefficients.

Our normalization factors $\sqrt{\frac{l+1}{\pi}}$ and $\sqrt{\frac{l+1}{2\pi}}$ are inspired by the common SH normalization factor. Koenderink et al. [1996] used a different one, namely $\sqrt{\frac{2l+1}{2}}$ for all three cases.

Domains Zernike's basis is defined on the disc, but can be mapped to the hemisphere.

Quality See Section 4.1 to ringing. Other than HSH, this basis can have different values on its equator and is still orthogonal.

Performance A lookup takes $\mathcal{O}(n^2)$ steps with n being the number of bands. However, the constant factor is smaller than for the other polynomial bases allowing slightly more bands for the same memory and computation requirements.

```
evalzernike(dir) -> s
    s, idx = 0
    r = sqrt(1-dir.z) # == sqrt(2) * sin(acos(dir.z) / 2)
    phi = atan2(dir.y, dir.x)
    # Iterate over m in the outer loop to reduce sin/cos calls.
    for m in [0, n-1]:
        sinMPhi = sin(m * phi)
        cosMPhi = cos(m * phi)
        # Enumerate bands which contain bases with index m.  /2*2 rounds to
        # the next smaller even number.  Alternatively use (n-m+1)&-2.
        for l in [m, +2, ((n-m+1)/2)*2 + m]:
            # Since Rlm and the normalization are equal for two bases m
            # and -m they can be factored out.
            Rlm_x = 0
            for k in [0,(l-m)/2]:
                Rlm_x += pow(r, l - 2 * k) * RK[idx++]
            end
            if m == 0:
                s += c[l * l / 2 + l] * Rlm_x
            else
```

```
                s += (c[(l * l - m) / 2 + l] * sinMPhi
                    + c[(l * l + m) / 2 + l] * cosMPhi) * Rlm_x
            end
        end
    end
end
```

Listing 10. Evaluation of Zernike basis. The array `RK` contains the fraction from Equation 16 which is independent of x times the normalization factor.

4.6. Makhotkin's Basis

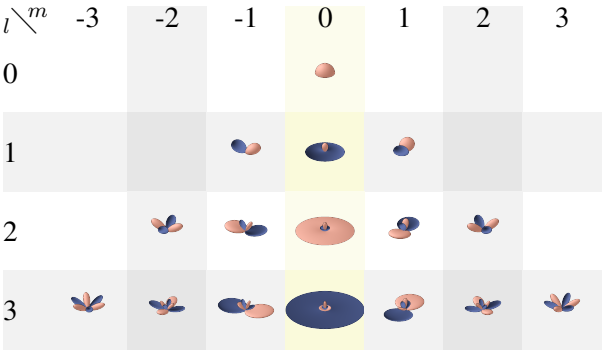


Figure 17. Makhotkin basis functions for the first four bands. Red and blue shades denote the sign. This basis has a severe problem with horizontal values as visible in column 0.

Makhotkin established another hemispherical basis to describe incoming and outgoing radiance on a surface [Makhotkin 1996]. He started with the Jacobi polynomials

$$J_l(x,0,1) = \frac{(-1)^n}{2^n n!(1+x)} \frac{\mathrm{d}^n}{\mathrm{d}x^n} \left[(1+x)(1-x^2)^n\right]$$

as orthogonal base. It should be mentioned the Legendre polynomials, used in SH basis, are another simpler special case of the Jacobi polynomials: $J_l(x,0,0)$, without the two $(1+x)$ terms. This leads to problems in orthogonality as pointed out below.

In the next step, Makhotkin defined the adjoint Jacobi functions

$$J_l^m(x) = (1-x^2)^{m/2} \frac{\mathrm{d}^n}{\mathrm{d}x^n} J_l(x,0,1)$$

which can also be described by the following recursive formulation:

$$\begin{aligned}
J_0^0(x) &= 1 \\
J_l^m(x) &= 0 && \text{if } l < m \wedge m < 0 \\
J_l^m(x) &= \frac{a_1 x - 1}{a_0} J_{l-1}^m(x) - \frac{a_2}{a_0} J_{l-2}^m(x) + \frac{a_1}{a_0} m (1-x^2)^{m/2} J_{l-1}^{m-1}(x)
\end{aligned}$$

$$a_0 = (2l-1)(l+1) \qquad a_1 = (2l-1)(2l+1) \qquad a_2 = (l-1)(2l+1)$$

Finally the hemispherical basis is:

$$M_l^m(\theta,\varphi) = \begin{cases} C_l^{|m|}\sin(|m|\varphi)\cdot J_l^{|m|}(2\cos\theta-1) & m>0 \\ C_l^{|m|}\cos(|m|\varphi)\cdot J_l^{|m|}(2\cos\theta-1) & m<=0 \end{cases}$$

Unfortunately, the normalization does not have a known closed form solution. We required that the square norm equals 1 yielding the following integral formulation:

$$\begin{aligned} C_l^m &= \sqrt{\frac{1}{\int_0^{2\pi}\int_0^{\pi/2}\left(J_l^m(2\cos\theta-1)cos(m\varphi)\right)^2\sin\theta\mathrm{d}\theta\mathrm{d}\varphi}} \\ &= \sqrt{\frac{2}{\int_0^{2\pi}\int_{-1}^{1}\left(J_l^m(x)cos(m\varphi)\right)^2\mathrm{d}x\mathrm{d}\varphi}} \end{aligned}$$

where the second form results from the substitution of $2\cos\theta-1$ with x. For the first 5 bands this gives:

$$\begin{array}{lllll} C_0^0 = \sqrt{\frac{1}{2\pi}} & & & & \\ C_1^0 = \sqrt{\frac{2}{4\pi}} & C_1^1 = \sqrt{\frac{2}{3\pi}} & & & \\ C_2^0 = \sqrt{\frac{3}{6\pi}} & C_2^1 = \sqrt{\frac{3}{12\pi}} & C_2^2 = \sqrt{\frac{3}{40\pi}} & & \\ C_3^0 = \sqrt{\frac{4}{8\pi}} & C_3^1 = \sqrt{\frac{4}{30\pi}} & C_3^2 = \sqrt{\frac{4}{240\pi}} & C_3^3 = \sqrt{\frac{4}{1260\pi}} & \\ C_4^0 = \sqrt{\frac{5}{10\pi}} & C_4^1 = \sqrt{\frac{5}{60\pi}} & C_4^2 = \sqrt{\frac{5}{840\pi}} & C_4^3 = \sqrt{\frac{5}{10080\pi}} & C_4^4 = \sqrt{\frac{5}{72576\pi}} \end{array}$$

Unfortunately, the basis is only orthogonal with respect to the weight function $1+x$, because the underlying Jacobi polynomials fulfill the condition

$$\int_{-1}^{1} J_{l_a}^{m_a}(x)J_{l_b}^{m_b}(x)(1+x)\mathrm{d}x = 0 \qquad \text{if } l_a \neq l_b \vee m_a \neq m_b$$

only [Makhotkin 1996]. This is not the case for Legendre polyonomials, which are orthogonal under the weight function $w(x)=1$. However, since all other terms in M do not depend on x we can orthogonalize the basis by multiplying with $\sqrt{1+x}$, yielding

$$\widetilde{M_l^m}(\theta,\varphi) = \begin{cases} C_l^{|m|}\sin(|m|\varphi)\cdot J_l^{|m|}(2\cos\theta-1)\cdot\sqrt{2\cos\theta} & m>0 \\ C_l^{|m|}\cos(|m|\varphi)\cdot J_l^{|m|}(2\cos\theta-1)\cdot\sqrt{2\cos\theta} & m<=0 \end{cases}.$$

The modification forces the values on equator to be zero, which is different to all other bases in this chapter. The newly obtained orthogonal basis performs much better than the original, as shown in supplemental 1.

Domains The Makhotkin basis is defined on the hemisphere.

Quality Except ringing (Section 4.1) this basis has severe problems with horizontal values and is not orthogonal in its original form. Like HSH, the equator can have only one constant value. We introduced an orthogonalized alternative which has zero value on its equator.

Performance A lookup is possible in $\mathcal{O}(n^2)$ steps with n being the number of bands.

5. Mixture Models

A component in our sense is a bell-shaped function on a sphere. A mixture model combines several of these components as a non-orthogonal base to approximate a function. The number of degrees of freedom and the used shape for each function can differ. Most common are Gaussian, Cosine and Beckmann base functions. The degrees of freedom (DOF) are the center direction of a lobe (2), an isotropic (1) or anisotropic (3) frequency and the amplitude (1). This gives up to 6 DOF per component. From those only the amplitude can be determined using linear optimization. The remaining DOFs must be found using iterative methods. Thereby, the most difficult part is to find well defined starting values to avoid local maxima.

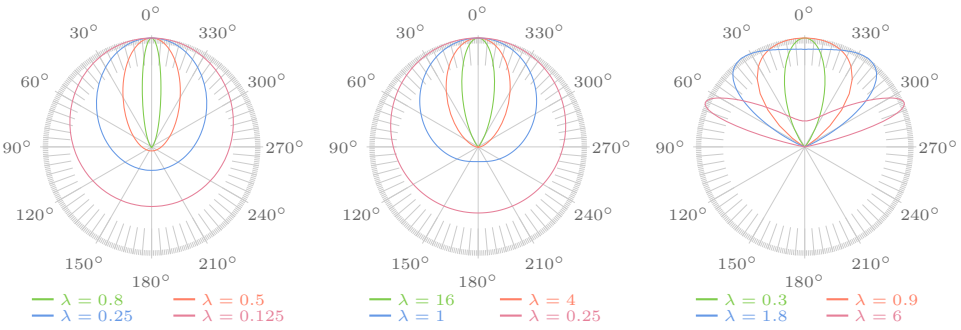
The reminder of this section is structured as follows: First, a list of common radial basis functions is introduced. Then, it is shown how to make any of the given kernels anisotropic. In Section 5.3 amplitudes are fitted in a least squares sense while keeping all other parameter fixed. As a first full fitting method we apply the non-linear solvers *Levmar* and *Ceres* and combine them with the linear determination of amplitudes. Section 5.5 introduces the Expectation Maximization algorithm and shows the parameter estimation for many of the introduced radial basis functions. Finally, an iterative approach is described which reduces the number of lobes until the desired count remains (Section 5.6).

5.1. Spherical Radial Basis Functions (SRBF)

An SRBF is a positive definite function G defined on the geodesic distance between two points on the m-sphere $\vec{c}, \vec{d} \in \mathbb{S}^m$: $\theta = \arccos(\vec{c} \cdot \vec{d})$. The SRBF is symmetric (isotropic) with respect to the central axis $\vec{c}$. A function f can be approximated as

$$f(\vec{d}) \approx \sum_{i=1}^{n} w_i G(\vec{c_i} \cdot \vec{d}, \lambda_i) \tag{18}$$

where w_k are weights (amplitudes) of single bases, c_i are the center directions of the components and λ_i their frequency parameter. Often SRBF are used in a context where only w_i are fitted to the function with the remaining parameters fixed upfront.



(a) *Abel-Poisson* **(b)** *von-Mises-Fisher* **(c)** *Beckmann*

Figure 18. Examples of SRBF kernels D with scaling to one of the maximum value for visualization purposes.

Section 5.3 handles this in detail. For varying center and frequency parameters, refer to Sections 5.4, 5.5 and 5.6.

In the following we give a list of possible functions with different normalization conditions. The normalization for probability distribution functions D on the sphere is $\int_\Omega D(\omega, \lambda)\mathrm{d}\omega = 1$. In a second normalization we require the zenith value to be one ($G(\vec{c} \cdot \vec{d} = 1) = 1$). This simpler condition is sufficient for some of the fit methods because multiplications by a scalar which does not depend on the spherical angle are consumed by w_k. In Figure 18 examples are visualized.

	D	G	λ
Beckmann	$\frac{e^{-(\tan^2\theta)/\lambda^2}}{\pi\lambda^2\cos^3\theta}$	$e^{-(\tan^2\theta)/\lambda^2}$	$(0,\infty)$
GGX	$\frac{\lambda^2\cos\theta}{\pi((\lambda^2-1)\cos^2\theta+1)^2}$	$\frac{\lambda^2}{((\lambda^2-1)\cos^2\theta+1)^2}$	$(0,1]$
Cosine	$\frac{\lambda+1}{2\pi}\cos^\lambda\theta$	$\cos^\lambda\theta$	$[0,\infty)$
vMF	$\frac{\lambda}{2\pi(e^\lambda-e^{-\lambda})}e^{\lambda\cos\theta}$	$e^{-\lambda}e^{\lambda\cos\theta}$	$(0,\infty)$
Gaussian A	$\frac{\lambda\cos\theta}{\pi(1-e^{-\lambda})}e^{-\lambda\sin^2\theta}$	$\cos\theta e^{-\lambda\sin^2\theta}$	$[0,\infty)$
Abel-Poisson	$\frac{1-\lambda^2}{4\pi(\lambda^2-2\lambda\cos\theta+1)^{3/2}}$	$\frac{(1-\lambda)^3}{(1-2\lambda\cos\theta+\lambda^2)^{3/2}}$	$(0,1)$

The actual Beckmann [Beckmann and Spizzichino 1987] and GGX [Walter et al. 2007] distributions are defined as normal distribution functions with a different normalization condition ($\int_\Omega D(\omega, \lambda)\cos\theta\mathrm{d}\omega = 1$). We derived the density distributions

by multiplying with $\cos\theta$. In case of D^{Beck} the maximum is not at $\theta = 0$ for $\lambda > \sqrt{2/3}$ as visible if Figure 18c. Our simplified G^{Beck} merely inherits the tangents distance and discards the entire denominator.

The cosine distribution is fairly known as Phong lobe in computer graphics and often used in reflection models.

The von-Mises-Fisher [Fisher 1953] distribution is one widely used form of a Gaussian kernel on the sphere. We added a further Gaussian variant with a different dependency on θ. This Gaussian A kernel is inspired by the Anisotropic-Spherical-Gaussians [Xu et al. 2013].

Finally, we found the Abel-Poisson distribution in [Freeden et al. 1997] and [Narcowich and Ward 1996] which has some nice properties opposed to other kernels in the list.

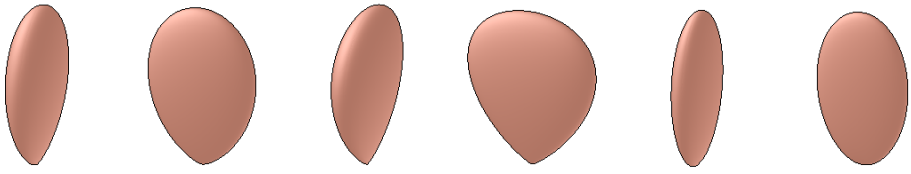
Only the Abel-Poisson and von-Mises-Fisher distributions span the entire sphere. All others are antipodal symmetric and are usually clamped to the positive hemisphere. I.e. even though not explicitly written the cosine terms must be surrounded with $\max(0, \cos\theta)$ in those kernels.

For two of the kernels closed form convolutions are given by Narcowich and Ward [1996]. In case of Abel-Poisson distribution the convolution yields another Abel-Poisson distribution again. The second one is a convolution of a Gaussian kernel in the form $e^{\lambda\cos\theta}$. The convolution is possible on the m-sphere, but only shown for the 2-sphere here. For an m-sphere formulation the reader is referred to [Narcowich and Ward 1996].

$$G^{\text{AP}} * H^{\text{AP}}(\vec{c}_g \cdot \vec{c}_h, \lambda_g, \lambda_h) = \frac{(1-\lambda_g\lambda_h)^3}{(1-2\lambda_g\lambda_h(\vec{c}_g \cdot \vec{c}_h) + (\lambda_g\lambda_h)^2)^{3/2}}$$

$$G^{\text{Gau}} * H^{\text{Gau}}(\vec{c}_g \cdot \vec{c}_h, \lambda_g, \lambda_h) = 4\pi \frac{\sinh\left(\sqrt{\lambda_h^2 + \lambda_g^2 + 2\lambda_h\lambda_g(\vec{c}_g \cdot \vec{c}_h)}\right)}{\sqrt{\lambda_h^2 + \lambda_g^2 + 2\lambda_h\lambda_g(\vec{c}_g \cdot \vec{c}_h)}}$$

5.2. Anisotropic SRBF



(a) $D^{\circlearrowleft Cos}$ *with* $\alpha = 3$, $\beta = 20$. **(b)** $D^{\circlearrowleft Beck}$ *with* $\alpha = 2$, $\beta = 10$. **(c)** $D^{\circlearrowleft GGX}$ *with* $\alpha = 2$, $\beta = 10$.

Figure 19. Examples of anisotropic kernels. Each of the functions is shown from two directions. Note that exponents of the Cosine distribution are greater to achieve a similar shape.

The SRBF kernels above all used the cosine of the angle θ between distribution center and the current direction. For anisotropic models the frequency λ must change according to the azimuth φ, too. To incorporate this we use two frequency parameters α and β in connection with the tangent $\vec{t}$ and the bitangent $\vec{b}$ of the central vector $\vec{c}$. Therewith $\vec{t}, \vec{b}$ and $\vec{c}$ form an orthonormal basis $\mathbf{L}$. Then, the exponent in the isotropic models can be replaced with

$$\lambda' = \frac{\alpha(\vec{t}\cdot\vec{d})^2 + \beta(\vec{b}\cdot\vec{d})^2}{1-(\vec{c}\cdot\vec{d})^2} = \frac{\alpha(\vec{t}\cdot\vec{d})^2 + \beta(\vec{b}\cdot\vec{d})^2}{(\vec{t}\cdot\vec{d})^2 + (\vec{b}\cdot\vec{d})^2} = \alpha\cos^2\phi + \beta\sin^2\phi. \quad (19)$$

If $\alpha = \beta$ Equation 19 simplifies to a single scalar due to orthonormality $(\vec{t}\cdot\vec{d})^2 + (\vec{b}\cdot\vec{d})^2 + (\vec{c}\cdot\vec{d})^2 = 1$. Care must be taken for the normalization terms which are now dependent on α and β.

This form of exponent is used in anisotropic BRDFs like Ashikhmin-Shirley [Ashikhmin and Shirley 2000] and Ward [Ward 1992], too. Inserting the new exponent in the isotropic kernels and finding new normalization conditions produces the following PDFs:

$$\begin{aligned}
D^{\circlearrowleft\text{Beck}} &= \frac{\sqrt{\alpha\beta}e^{-(\tan^2\theta)(\alpha\cos^2\phi+\beta\sin^2\phi)}}{\pi\cos^3\theta} \\
D^{\circlearrowleft\text{GGX}} &= \frac{\sqrt{\alpha\beta}}{\pi(\cos^2\theta+\alpha\cos^2\phi+\beta\sin^2\phi)^2} \\
D^{\circlearrowleft\text{Cos}} &= \frac{\sqrt{(\alpha+1)(\beta+1)}}{2\pi}\cos\theta^{\alpha\cos^2\phi+\beta\sin^2\phi} \\
D^{\text{Kent}} &= \frac{1}{2\pi\sum_{j=0}^{\infty}\frac{\Gamma(j+\frac{1}{2})}{\Gamma(j+1)}\beta^{2j}\left(\frac{\alpha}{2}\right)^{-2j-\frac{1}{2}}I_{2j+\frac{1}{2}}(\alpha)}e^{\alpha\cos\theta+\beta((\vec{b}\cdot\vec{d})^2-(\vec{t}\cdot\vec{d})^2)} \\
D^{\mathbb{C}\text{B}} &= \left(2\pi^k\sum_{j=1}^{k}\frac{e^{\lambda_j}}{\prod_{i!=j}(\lambda_j-\lambda_i)}\right)^{-1}e^{\vec{d}^H\mathbf{A}\vec{d}} \\
D^{\mathbb{C}\text{K}} &= \frac{\Gamma(k)}{2\pi^k}|\mathbf{K}|^{-1}(\vec{d}^H\mathbf{K}^{-1}\vec{d})^{-k} \\
G^{\text{ASG}} &= \cos\theta e^{-\alpha(\vec{t}\cdot\vec{d})^2-\beta(\vec{b}\cdot\vec{d})^2}
\end{aligned}$$

Thereby D^{Kent} is the Kent distribution [Kent 1982] which requires the gamma function Γ and the modified Bessel function I_x for normalization. It is related to the von-Mises-Fisher distribution D^{vMF} which is the special case of the Kent distribution for $\beta = 0$. Here, $\alpha > 0$ determines the concentration and $\beta \in [0, \alpha/2)$ the degree of anisotropy which is different from our Equation 19.

Two other distribution introduced by Kent [1994; 1997] are the complex Bingham distribution $D^{\mathbb{C}\mathrm{B}}$ and some other complex distribution we call $D^{\mathbb{C}\mathrm{K}}$ (complex Kent). The matrices $\mathbf{A}$ and $\mathbf{K}$ defines their shapes. Let λ_i be the eigenvalues of $\mathbf{A}$ which determine the normalization of $D^{\mathbb{C}\mathrm{B}}$. Both distributions are defined for complex unit spheres in k (in our case $k = 3$) dimensions and allow $\vec{d}$ to be a k dimensional complex vector. For that reason the conjugate transpose $\vec{d}^H$ is used instead the usual transpose $\vec{d}^T$.

G^{ASG} is the aforementioned Anisotropic-Spherical-Gaussian function from Xu et al. [2013]. We were not able to find the the PDF normalization which is likely similar to that of the Kent distribution.

5.2.1. *Tangent Space of Anisotropic Kernels*

For a given $\vec{c}$ the choice of tangent and bitangent requires an additional parameter of rotation ϕ to fix the remaining DOF. The orthogonal space is represented uniquely with

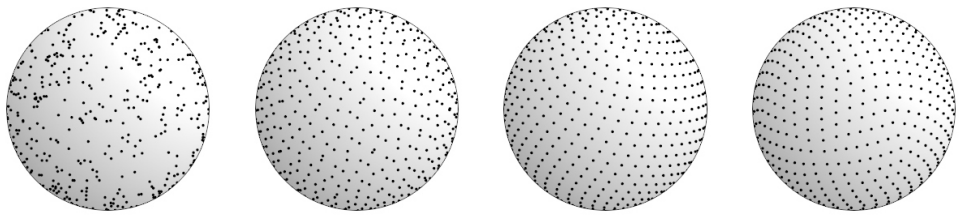
$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \cdot \begin{bmatrix} \sin\theta\cos\varphi & \sin\theta\sin\varphi & \cos\theta \\ -\sin\varphi & \cos\varphi & 0 \\ \cos\theta\cos\varphi & \cos\theta\sin\varphi & -\sin\theta \end{bmatrix}$$

such that

$$(\vec{c}\cdot\vec{d}, \vec{t}\cdot\vec{d}, \vec{b}\cdot\vec{d})^T = \mathbf{L}\cdot\vec{d}.$$

I.e. the rows of matrix $\mathbf{L}$ are the vectors $\vec{c}, \vec{t}$ and $\vec{b}$ using the convention from Section 2.3 that z is the up axis.

5.3. SRBF with Fixed Centers and Frequencies



(a) *Random* **(b)** *Hammersley* **(c)** *Golden Ratio* **(d)** *HEALPix*

Figure 20. Center distributions for $n = 768$.

This section demonstrates the linear optimization of amplitude weights using fixed centers c_i and frequency parameters λ_i or α_i, β_i.

Narcowich and Ward [1996] introduced SRBF for continuous wavelet transformations on the m-sphere. They describe a mathematical foundation and show how to localize the wavelets G. To setup the centers c_i, they proposed to use subdivisions

of the icosahedron, but did not detail this choice. Their work inspired the following applications of SRBF in computer graphics.

SRBFs were also used for PRT lighting in [Tsai and Shih 2006]. For the light transfer function they placed the centers c_i on a subdivided icosahedron as proposed by Narcowich and Ward [1996]. The resulting matrix was then compressed using Clustered PCA (Principal Component Analysis). To compress the environment map they used scattered SRBFs, meaning that all parameters of the mixture model are optimized. This general optimization is detailed in the next sections.

Leung et al. [2006] used SRBFs for image based lighting. They stated this basis is faster than SH because all bases have the same shape. On the other hand it looses the orthogonality and still shows ringing artifacts. Leung at al. used a Hammersley point (See Figure 20b) set to generate center directions c_i and used the same λ_i for all components. λ_i can be chosen dependent on the smallest geodesic distance between the center points.

The SRBF can be fitted using least squares or regularized least squares as in [Leung et al. 2006]. For n kernels with centers $\vec{c_i}$, m samples with directions $\vec{d_i}$ and values $b_i = f(\vec{d_i})$ the weights w_i are obtained by solving

$$\mathbf{A}\vec{w} = \vec{b}$$

$$\mathbf{A} = \begin{bmatrix} G_1(\vec{c_1} \cdot \vec{d_1}) & \cdots & G_n(\vec{c_n} \cdot \vec{d_1}) \\ \vdots & \ddots & \vdots \\ G_1(\vec{c_1} \cdot \vec{d_m}) & \cdots & G_n(\vec{c_n} \cdot \vec{d_m}) \end{bmatrix}.$$

For this overdetermined equation system, the unconstrained solution is

$$\vec{w} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\vec{b}$$

and the regularized variant is

$$\vec{w} = (\mathbf{A}^T\mathbf{A} + \varepsilon\mathbf{I}_{n\times n})^{-1}\mathbf{A}^T\vec{b}$$

with $\mathbf{I}_{n\times n}$ being the identity matrix.

Reasonable distributions for the centers c_i are shown in Figure 20. Using random numbers produces poor results due to high discrepancy. Two other common choices are Hammersley point sets and the pixel centers of grid based models. To generate a Hammersley point set the number of lobes n must be given in advance. n can be chosen arbitrary which allows a very fine control of how many lobes with a certain λ should be generated. However, the subdivision of grids allows adaptive tessellation to setup components with different λ values.

We propose another low-discrepancy series, titled with 'Golden Ratio' in Figure 20c. It can be initialized for any number n like the Hammersley set, but is faster to

compute and better distributed. It uses the two sequences

$$\xi_i = \frac{i}{n} \qquad \text{and} \qquad \zeta_i = \Phi \cdot i \mod 1 \tag{20}$$

with the golden ratio $\Phi = (1+\sqrt{5})/2$. The second sequence is known as the additive recurrence generator with the smallest possible discrepancy [Mollwollfumble 2011]. Hammersley sets use the reversed binary representation of ξ_i instead, which is more difficult to obtain. In both cases the two numbers ξ_i and ζ_i are transformed to uniform distributed directions on the sphere using:

$$\theta = \arccos(1 - 2\xi_i) \qquad \varphi = 2\pi\zeta_i$$

To determine λ the average or minimal angle ω between two of the generated centers can be inserted into

$$G(\cos\frac{\omega}{2}, \lambda) = a.$$

and solved for λ. Here, a is a parameter of how much contribution a kernel should have in the midsection of two kernels. We found $a \in [\frac{1}{5}, \frac{1}{3}]$ to perform best where smaller numbers were better in functions with large peak values (HDRdat).

Domains Radial basis functions are defined on any dimensional sphere $\mathbb{S}^d$. They can be used for hemispheres or even arbitrary sphere sectors by modifying the sampling set.

Quality SRBF show a ringing-like pattern, but other than polynomial bases there is no repeating of features or overshooting. I.e. there is no ringing in the classical sense. Further, the quality depends on the uniformity of sample distribution which can be ensured easily using Equation 20.

SRBF are rotational invariant by simply rotating center directions.

Performance A lookup takes $\mathcal{O}(n)$ for n components. To fit the weights $\mathcal{O}(n^3)$ steps are necessary to solve the least squares problem.

5.4. Fitting through Non-linear Solvers

Commonly, lobe mixture models are variable in all parameters including the orientation $(\vec{c}, \vec{t}, \vec{b})$ and the exponents (λ or α, β). The previous section showed that the weight for each lobe can be optimized in a least squares sense linearly. This is not possible for the other parameters.

We know of three different approaches to fit all the parameters: non-linear optimization, expectation maximization and iterative constructions. The first method is the topic of this section, the other two follow in the next sections. Each of the methods requires an initial guess for the parameters. Dependent on this initialization the methods will converge into local optima.

Using a non-linear gradient decent solver is the easiest but least robust solution. It can be applied for any isotropic or anisotropic kernel. The Levenberg-Marquardt optimization algorithm [Levenberg 1944; Marquardt 1963] is often used in this context. An implementation is freely available in the *Levmar* library [M.I.A. Lourakis 2004]. Another open source optimization library is *Ceres* [Agarwal et al. 2010] which is based on different algorithms including Levenberg-Marquardt.

Both libraries require a number of residuals $f(\vec{x}) - \mathrm{mm}(\vec{x}, \vec{p})$ where f is the original function and mm is the mixture model using parametrization $\vec{p}$. The number of parameters to fit is either four (isotropic) or six (anisotropic) times the number of lobes plus one. We added the one extra value as constant offset over the entire sphere.

We tested both with respect to the robustness against local optima. Therefore, we computed the statistics for multiple runs over the test data set with different initializations. A smaller variance reveals a higher robustness for the observed fits independent on the initialization.

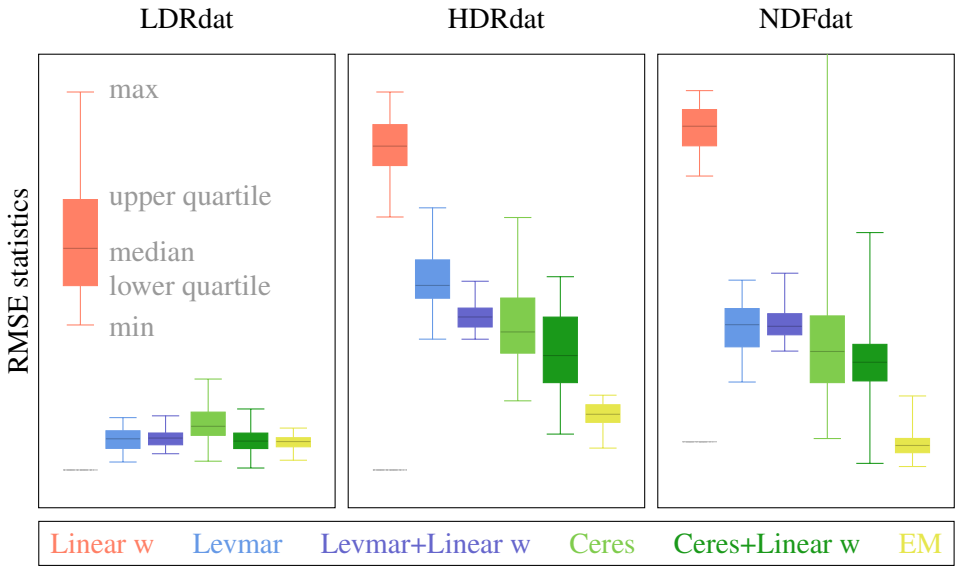


Figure 21. Performance of *Levmar* and *Ceres* non-linear optimization libraries and expectation maximization algorithm. A 16 component isotropic cosine model was fitted to the test data. Each solver was executed 12 times per file with different random initializations to provide minimum, maximum, median and the two quartiles of RMSE. Each boxplot shows the geometric mean $(\prod x_i)^{1/n}$ computed over all files in a data set. Smaller values represent better fits, whereas compacter ranges show a higher robustness. The single line on the bottom left of each plot is the deterministic mixture-reduction method from Section 5.6 followed by EM.

Figure 21 presents the statistic over all data sets for a fitting of 16 isotropic cosine kernels. As reference we used the linear optimization of weights from the previous

section. It should always perform worst, because directions and exponents are not fitted. Then we used both libraries in two different modes. First by executing the non-linear optimization for all parameters and second by interleaving iterations for directions+exponents with linear optimization of the weights.

The interleaved approach is more robust and produces better results when using *Ceres*. Using the *Levmar* library both techniques produce similar outputs. Hence, interleaving iterations should be preferred. In comparison between the libraries *levmar* seems to be more robust than *Ceres*. However, *Ceres* often produces the better fits. It also provides a lot of options which we did not test completely. Therefore, *Ceres* could be configured more robustly.

With respect to run time both libraries are similar, if using analytical Jacobian matrices. When using finite differences *Levmar* is much faster. In all cases a fit requires several minutes and execution time varies over the data sets (1-15 min). In comparison, the linear fitting of the weights takes half a second.

5.5. Fitting through Expectation Maximization

The EM algorithm [Dempster et al. 1977; Bilmes and others 1998] is an iterative method, designed to find the parameters for a number of independent probability distributions such that their sum fits a target probability distribution function (PDF). Without normalizing the integral to one, any function can be approximated as a sum of functions.

The algorithm consists of two steps. In the expectation step (E-Step) the probability to observe a certain sample under a specific model is determined. Given that probability, a weighted assignment of samples to components is made. In the maximization step (M-Step) the parameters of the distribution functions are recalculated to maximize the likelihood for the assigned data.

The two steps can be applied in two different ways: off-line-EM and on-line-EM. In the off-line version all assignments are computed first, before all components are updated. This two phases are then repeated iteratively. In contrast, the models are updated immediately after observing a single sample in on-line methods. This allows to reject samples after their observation. We implemented the off-line method, but the following estimations are valid for both versions.

From all the spherical kernels we employed above, the von-Mises-Fisher distribution is the most famous one in connection with EM. Banerjee et al. [2005] introduce the EM algorithm for vMF to cluster data on high dimensional unit spheres. There is also a library implementation in R and a comparison to spherical k-means in [Hornik and Grün 2014].

$i \in [1, N]$	Index of the kernel or reference to its parameter set
$s \in [1, M]$	Index of the sample
$\vec{d}_s, f_s$	Observed direction and function value of sample s
$\vec{c}_i$	Central distribution direction; often μ in EM descriptions
θ_i	Isotropic or anisotropic frequency parameter of component i
w_i	Scale of a component (weight of the basis function)

In order to fit arbitrary functions each sample has a weight f_s. Intuitively this can be seen as the number of observed samples in that direction. The same approach was applied in [Vorba et al. 2014], which also provides more detailed derivations. If f_s is removed from all following equations a standard EM formulation is obtained.

5.5.1. E-Step

The probability that a sample is produced from a kernel i is

$$p(\vec{d}_s|i) = \frac{G(\vec{d}_s \cdot \vec{c}_i, \theta_i)}{\int_\Omega G(\vec{d}_s \cdot \vec{c}_i, \theta_i) \mathrm{d}\omega}$$
$$\text{and} \qquad h_{si} = \frac{w_i p(\vec{d}_s|i)}{\sum_{k=1}^{N} w_i p(\vec{d}_s|k)} \tag{21}$$

its assignment to that kernel. If the kernel G already is a probability density function the normalization integral vanishes. With (21) a weighted statistic of sample directions can be calculated by

$$(n_i, \vec{\mu}_i, \mathbf{S}_i) = \frac{1}{M} \sum_{s=1}^{M} f_s h_{si} \cdot (1, \vec{d}_s, \vec{d}_s \vec{d}_s^T)$$

5.5.2. M-Step

According to Buchta et al. [2012] the average of unit vectors $\vec{\mu}_i$ maximizes the cosine similarity $\sum_s f_s h_{si} w_i \langle \vec{d}_s, \vec{c}_i \rangle$ between the kernel direction and all assigned samples. Therefore, the update of the new central directions is straight forward. It is the normalized direction of the weighted average

$$\vec{c}_i^{\text{new}} = \frac{\vec{\mu}_i}{\|\vec{\mu}_i\|}.$$

The estimation of the kernel scales w_i is essential the linear fitting problem from Section 5.3 in the general. However, usual EM formulations ([Bilmes and others 1998]) provide the following estimation:

$$w_i^{\text{new}} = \frac{\sum_s h_{si}}{\sum_{k=1}^{N} \sum_s h_{sk}},$$

which is based on the additional property $\sum_i w_i = 1$. We can extend this formula for a weighted approach, still enforcing a sum of one for the kernel scales

$$w_i^{\text{new}} = \frac{\sum_s f_s h_{si}}{\sum_{k=1}^{N} \sum_s f_s h_{sk}} = \frac{n_i}{\sum_{k=1}^{N} n_k}.$$

Thus, the fitting procedure can be executed without linear optimization in the first place. Then, the final result must be corrected by scaling the results with $\int_\Omega f$ or by a final linear optimization.

The frequency parameter estimation is different for all kernels, but they can be derived with the same approach. A common target function to maximize the likelihood of a parameter set is the log-likelihood

$$\mathcal{L}(i) = \sum_{s=1}^{M} f_s h_{si} \log p(\vec{d}_s|i), \tag{22}$$

here extended by the weight f_s. Each parameter x can be optimized by maximizing $\mathcal{L}$ with respect to x by setting $\partial\mathcal{L}/\partial x = 0$. Following this idea we found solutions for D^{Cos} and D^{Beck}. Additionally, Banerjee et al. [2005] suggest an approximation for the von-Mises-Fisher distribution. Following their ideas, we found a similar approximation for the D^{GauA} function. The derivations of all our solutions can be found in the supplemental document 1.

$$\begin{aligned}
D^{\text{Cos}} \qquad & \lambda^{\text{new}} = \frac{-Mn_i}{\sum_s f_s h_{si} \log\cos\theta_s} - 1 \\
D^{\text{Beck}} \qquad & \lambda^{\text{new}} = \sqrt{\frac{\sum_s f_s h_{si} \tan^2\theta_s}{Mn_i}} \\
D^{\text{vMF}} \qquad & \lambda^{\text{new}} = \frac{r(3-r^2)}{1-r^2} \text{ with } r = \frac{\|\vec{\mu}\|}{n_i} \\
D^{\text{GauA}} \qquad & \lambda^{\text{new}} = \frac{1-8r^3}{r} \text{ with } r = \frac{\sum_s f_s h_{si} \sin^2\theta_s}{Mn_i}
\end{aligned}$$

5.5.3. EM for Anisotropic Models

The above approach of optimizing $\partial\mathcal{L}/\partial x = 0$ is still valid for anisotropic models. However, before the exponents can be estimated we require the tangent space directions. We found that eigenvectors of $\mathbf{S}_i$ correlate with our searched tangent space. The eigenvector corresponding to the largest eigenvalue is very close to $\vec{c}_i$. Hence, the other two eigenvectors yield approximations to tangent $\vec{t}_i$ and bitangent $\vec{b}_i$ after re-orthogonalization with respect to $\vec{c}_i$. We did not test to maximize the log-likelihood for a direction ϕ explicitly which could be possible too.

Once the tangent and bitangent are known the log-likelihood (Eq. 22) can be maximized for the exponent parameters. The optimization for the anisotropic parameters

of the cosine distributions produced:

$$\alpha^{\text{new}} = \frac{-Mn_i}{2\sum_s f_s h_{si} \cos^2\phi \log\cos\theta} - 1$$

and the result for the Beckmann distribution is:

$$\alpha^{\text{new}} = \frac{Mn_i}{2\sum_{s=1}^{M} f_s h_{si} \cos^2\phi \tan^2\theta}.$$

In both cases the β parameter is estimated equivalently by replacing $\cos^2\phi$ with $\sin^2\phi$ in front of the logarithm/tangents. The derivations are given in supplemental document 1 again. Similar to the isotropic case the estimation is more complicated for the other models and requires approximate solutions.

For the two distributions D^{CB} and D^{CK} Kent itself provides maximum likelihood estimates in [Kent 1994] and [Kent 1997]:

$$D^{\mathbb{C}\text{B}} \qquad \mathbf{A}_i^{\text{new}} = \mathbf{U}_i \operatorname{diag}(-\frac{1}{l_1}, ..., -\frac{1}{l_{k-1}}, 0)\mathbf{U}_i^H$$
$$\text{with } \mathbf{S}_i = \mathbf{U}_i \operatorname{diag}(l_1, ..., l_k)\mathbf{U}_i^H$$
$$D^{\mathbb{C}\text{K}} \qquad \mathbf{K}_i^{\text{new}} = \frac{k}{M}\sum_{s=1}^{M} \frac{\vec{d}_s \vec{d}_s^H}{\vec{d}_s^H \mathbf{K}_i^{-1} \vec{d}_s}$$

In the estimation of $\mathbf{A}$ the spectral decomposition of the statistics $\mathbf{S}$ is required. Therewith, the maximum eigenvalue of $\mathbf{A}$ is fixed to zero because $\mathbf{A}$ and $\mathbf{A} + \varepsilon\mathbf{I}$ describe the same distribution. The exponent H is the conjugate transpose again.

According to Kent the second distribution is more resistant against outliers. For more information to directional statistics please refer to [Mardia and Jupp 2000].

5.5.4. Maximum A-Posteriori Estimation

In some cases, especially in on-line EM implementations, over-fitting becomes a problem. It is possible to introduce a regularization by maximizing the posterior distribution $p(i|\mathcal{S})$ instead of the likelihood $p(\mathcal{S}|i)$. This can be modeled using Bayes' theorem as $p(i|\mathcal{S}) \propto p(\mathcal{S}|i)p(i)$, where $p(i)$ is the prior belief to obtain a certain parametrization. More details for MAP in the application of Gaussian mixture models can be found in [Gauvain and Lee 1994].

5.5.5. Variants of the EM Algorithm

Above, we mainly described the batch-EM variant which computes the statistics over all samples before estimating the new distribution parameters. In contrast, the online variant updates the parameters after each observed sample. Therefore, the sufficient statistics must be updated iteratively. In some literature ([Neal and Hinton 1998]) this is also called iterative EM. As shown in [Neal and Hinton 1998] this kind of online/iterative implementation converges faster, because each new information is included immediately and any further E-step is based on a more likely parameter set.

Further, Neal and Hinton comment on sparse EM as an additional performance optimization. This variant ignores certain samples for some of the distributions to avoid far ranged influences if there are enough closer components.

One of the most interesting applications of GMM in rendering, we found, is a grid of GMM for volumetric lighting [Jakob et al. 2011]. They use an EM-variant which they call progressive accelerated EM. This can be seen as a hierarchical refining sparse EM.

5.6. Fitting through Reduction and Refinement

Another approach to generate GMMs is to iteratively add or remove components from another GMM. We found several good overviews on reduction techniques [Crouse et al. 2011; Ardeshiri et al. 2015]. While there are many more involved algorithms we show only very fast methods to reduce models beginning with several thousands components. With such an method an initial distribution could be generated from the input cube maps with one component per pixel (millions) and then reduced down to the desired number of components. It happens that some of the greedy initialization methods of complex algorithms are already quite good and compete with the more complex algorithms itself. Crouse et al. [2011] state that Runnalls' method [Runnalls 2007] has similar quality like much more complex algorithms. Also Crouse et al. [2011] introduce West's [West 1993] greedy initialization.

All those algorithms are based on the standard multivariate (k-dimensional) normal distribution

$$\mathcal{N}[\Sigma, \vec{\mu}](\vec{x}) = \frac{1}{\sqrt{(2\pi)^{k/2}|\Sigma|}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})\right).$$

Merging two Gaussian kernels i and j with weights w is possible in a closed form using

$$\begin{aligned} w_{ij} &= w_i + w_j \\ \vec{\mu}s_{ij} &= \frac{w_i}{w_{ij}}\vec{\mu}_i + \frac{w_j}{w_{ij}}\vec{\mu}_j \\ \Sigma_{ij} &= \frac{1}{w_{ij}} \sum_{k\in\{i,j\}} w_k \left(\Sigma_k + (\vec{\mu}_k - \vec{\mu}_{ij})(\vec{\mu}_k - \vec{\mu}_{ij})^T\right) \end{aligned}$$

which is the foundation of all reduction algorithms.

In Runnalls' method each merge searches for the pair i, j that minimizes the Kullback-Leibler divergence. This measure describes the amount of information lost if one probability function is approximated by another. There is no closed form in case of the Gaussian kernels, but Runnalls gave an approximation for the upper bound of the KL divergence and derived the cost function

$$\text{cost} = \frac{1}{2}(w_{ij}\log|\Sigma_{ij}| - w_i\log|\Sigma_i| - w_j\log|\Sigma_j|).$$

Hence, Runnalls' algorithm has a quadratic runtime cost (for the pair-search) in each reduction step. Contrary, the method of West always takes i as the component with the lowest modified weight $w_i/\text{trace}(\Sigma_i)$ which we can obtain fast using a priority queue. Each step in the search takes linear time then. Moreover, West uses another cost function which minimizes the squared integral error

$$\int_x (f_i(x) - f_j(x))^2 \mathrm{d}x$$

namely

$$\text{cost} = w_i^2 \mathcal{N}[2\Sigma_i, \vec{\mu}_i](\vec{\mu}_i) + w_j^2 \mathcal{N}[2\Sigma_j, \vec{\mu}_j](\vec{\mu}_j) - 2w_i w_j \mathcal{N}[\Sigma_i + \Sigma_j, \vec{\mu}_i](\vec{\mu}_j).$$

We tried both search methods with both error functions and found Runnalls global search to produce the best results. However, using West's search with a fixed i together with the KL cost function produces similar results faster. The second cost function preformed worse for both algorithms. If reduction goes down to very few lobes (< 100) the quality of Runnalls method is worth the time spent.

Finally we used the following approach to fit the lobe models in the upcoming comparison section and for Figure 21.

1. Sample the target function at $n \cdot 20$ uniformly distributed locations, where n is the target number of components. (We used the deterministic Golden Ratio sampling as in Figure 20c.)

2. Run Runnalls reduction method down to n components. At any merge renormalize μ to keep the distributions on the sphere surface.

3. Use the above result to initialize the kernels (any model G).

4. Run the EM algorithm.

5. Find optimal weights w with linear optimization.

The factor 20 in the first step is an arbitrary factor determined by experimentation. It is a good compromise between number of components and approximation of the target function. Additionally, we modified the reduction algorithm by reprojecting $\vec{\mu}_{ij}$ onto the sphere. This forces the 3D multivariate Gaussian to be more similar to our other kernel models and thus produces better results if used as initialization. A comparison to the previous fitting methods is given in Figure 21 (single line in the bottom left of each plot). Only in few cases the random initialization led to better results and for the HDR data set the reduction technique performed superior to all other methods.

Domains Arbitrary sphere surface sectors.

Quality Mixture models result in smoothed versions of the input function with focus on high-energy regions. It is possible that only certain features are reproduced other than the general appearance, if over-fitting occurs.

The mixture models are rotational invariant by simply rotating center directions.

Performance A lookup takes $\mathcal{O}(n)$ for n components. In the above fitting procedure step 1 requires $\mathcal{O}(s^3)$ steps to reduce s samples down to n using Runnalls method. The batch-EM method requires $\mathcal{O}(n \cdot s)$ steps per iteration and often converges in less then 100 iterations. Since $s \gg 100 \wedge s > n$ the total running time is limited by $\mathcal{O}(s^3)$, which is the reason that we used only $s = n \cdot 20$ samples.

6. Model Comparison

In this section all previously introduced models are compared for their ability to match the target function relative to their space consumption. Further, we include a theoretical comparison of the fitting and sampling performance for each model. Providing measured timings is difficult, because of the many different influences. Some models depend on the size of the input data set and others scale with the increasing number of target basis functions. Further, our testing framework is designed for flexibility of different models and by far not optimal for most of them. Most certainly each of the models can be optimized with additional effort and therefore timing comparisons at the current state would not be reliable.

6.1. Spherical Functions

First we want to compare the fully spherical functions which include some of the hemispherical models (like projections) with two hemispheres. If applicable for both domains, the models do not behave very differently. The next section, for hemispherical models only, is mainly to compare the different polynomial bases to each other.

In the following each model is fitted against each data set using a number of different target parameters. Each set of parameters yields a model specific space consumption with all values being stored as 32-bit float values. While this is not optimal for LDR data and further optimizations would be possible by different discretizations it is still a fair comparison between the models. For reasons of time consumption and numerical issues, the number of tested parameters varies for different models. Finally, the errors inside each data set are averaged using the geometric mean $(\prod x_i)^{1/n}$ to produce a single graph for comparison.

Figure 22 shows the RMSE for all models in two different ways. On the left hand we can see a common plot. As expected, we can observe a monotonous decrease in error for all models. Since the graphs overlap strongly, the right side shows the errors

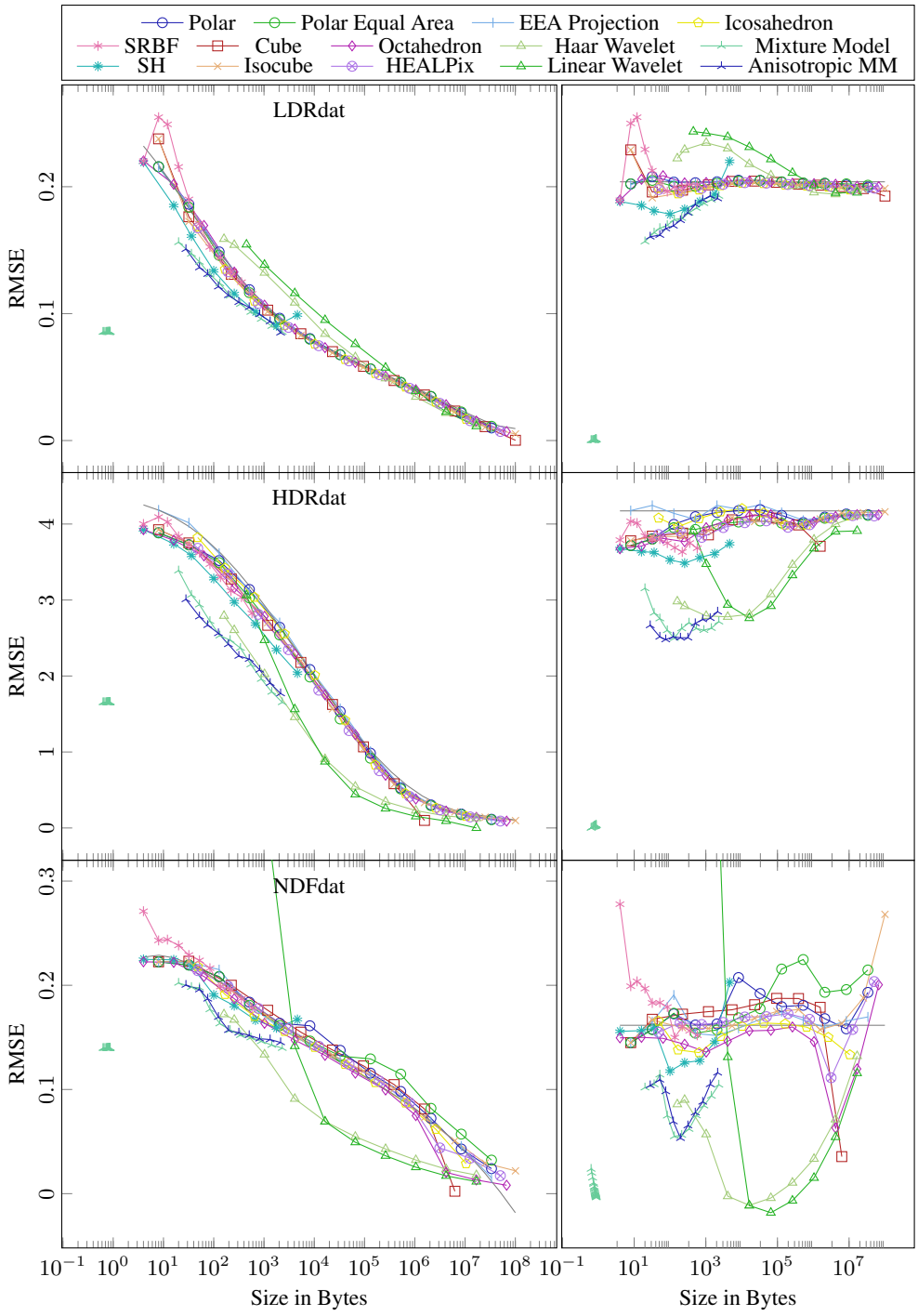


Figure 22. Error over size comparison for spherical models. On the right we subtracted offset polynomials to emphasis the differences between models. The polynomials are fitted to each data set individually and are visualized by the (straight) gray line.

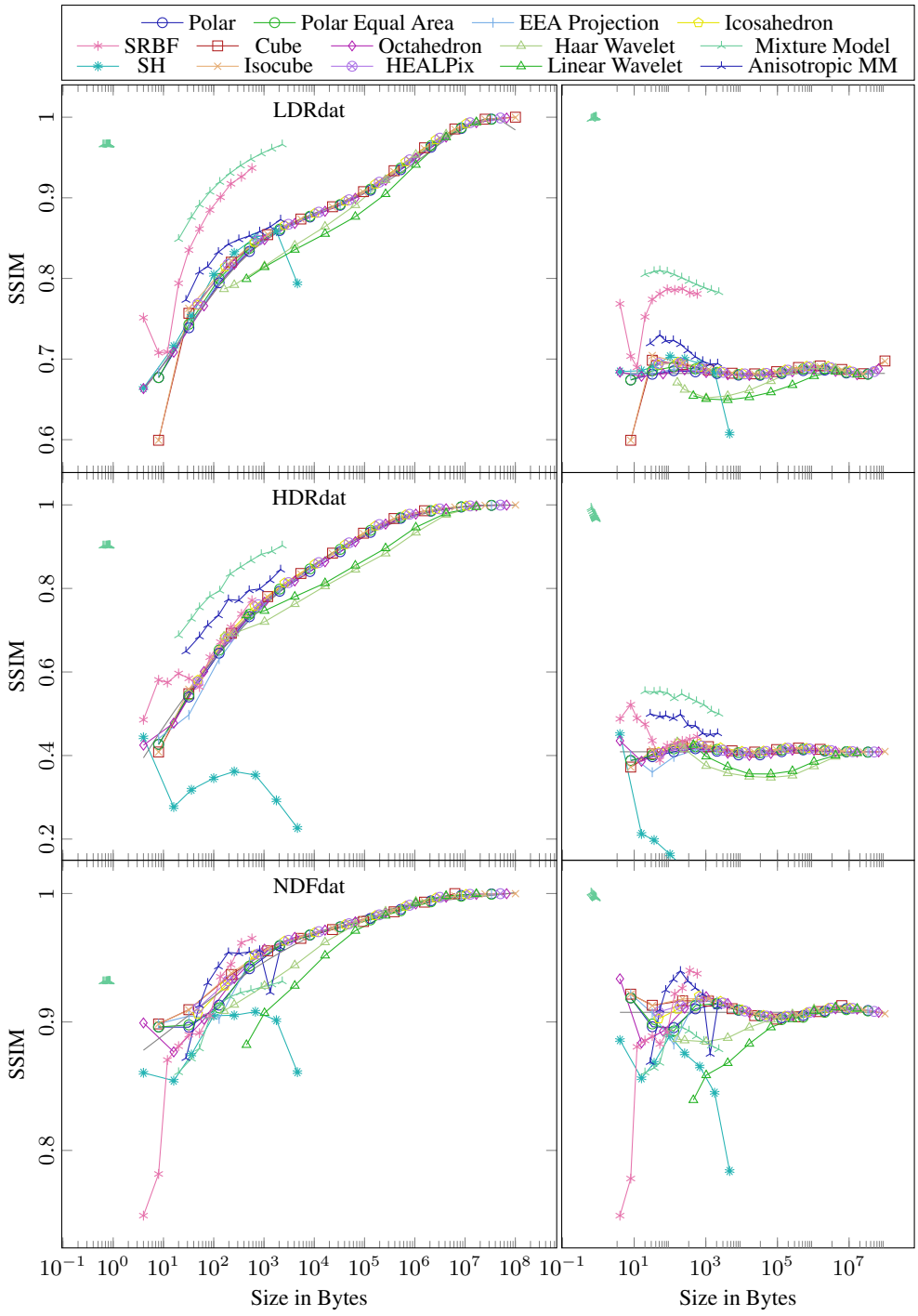


Figure 23. Structural similarity over size comparison for spherical models. On the right we subtracted offset polynomials to emphasis the differences between models. The polynomials are fitted to each data set individually and are visualized by the (straight) gray line.

relative to a fitted polynomial. While the slopes in this visualization have no meaning, it allows to compare the models against each other more easily. A second Figure (23) shows the structured similarity (SSIM) in the same way as Figure 22. Additionally, example images can be found in the supplemental document to get a visual impression of the models.

Most of the grid models perform very similar. Polar coordinates and projections are worse than many other models in many cases. For projection, only elliptical equal area mapping is shown, but the others did not produce lower errors. Since the input is provided as cube map, the cube map itself reaches zero error in the last sample. For reasons of pattern alignment it fails in the NDF data set over most sizes. In general, the more equally distributed models like HEALPix and Icosahedral maps seem to be more robust and produce good results for any data. Especially in polar coordinates, the alignment with the data is important and can influence the quality a lot. The SSIM graphs reflect the same observations for grid based models.

The much used SH basis is slightly better than the mapping based methods with respect to RMSE, but is more expensive than these. The curves are shown without windowing, which produces closer fits in the sense of RMSE. On the other hand, the distracting ringing has a high penalty on SSIM, where the SH is much worse than grid based models for HDR and NDF data. Damping the coefficients of higher bands visually improves the results (see Supplemental 2), but reduces the memory to error ratio in RMSE. With windowing the SH is not better than any grid method in least squares sense. In that case simple grids like EEA projection or octahedral mapping should be preferred.

In the long run the adaptive wavelet subdivision performs much better than anything else for HDR data in RMSE, while discontinuities lead to bad SSIM values. However, implementing the quad-tree on the sphere adds some overhead and redundancy in case of the linear vertex base. This overhead is too large in connection with uniformly distributed data like in the LDR test data. Further, the wavelets using adaptive coefficient trees cannot compete in the low memory range below 1000 Byte. Note that we did not use discretization or entropy encoding to make use of the value distributions in the coefficients. Additionally, it is also possible to implement the wavelets on top of any mapping method in the usual 2D domain.

If compression should go below 1000 Byte lobe mixture models are the clear winner and are recommended as long as time constraints do not prohibit the fitting process. At least for our data there is no real difference between anisotropic and isotropic components with respect to RMSE. The anisotropic components can fit some features better, but also need two additional parameters. Further, the perceived similarity is higher for isotropic fits, except for NDF data which indeed has an higher anisotropy. Therefore, the anisotropic form should only be considered, if the data is known to be anisotropic.

The alternative using fixed centers and exponents for the mixture components is not better than any mapping model in general. In some occasions it shows an higher similarity than most other models, but does not do so reliable. It might be more effective if some structure in the data is known. Then, directions and exponents could be fitted over all data sets. Amplitudes for each individual data can be computed with this data specific component distribution. This may achieve similar results as an unrestricted mixture model, if the data shares a common structure.

6.2. Hemispherical Functions

The experiments for hemispherical functions were performed in the same way as for spherical functions, except that only half of the data (positive z axis) was presented to the fitting algorithm and used in error computation. Overall, the behavior of the different models is very similar to previous comparison as shown in Figure 24.

From the grid based models, the octahedral mapping performs best with respect to both measurements. Overall, using mapping based methods seems a good choice in the hemispherical for many applications.

Again, wavelets and mixture functions have the best compression ratios for the same reasons as before. Particularly, mixture models perform exceptionally well in visual structure (SSIM), whereas wavelets are again penalized by discontinuities. An exception for RMSE is the mixture function in NDFdat. The poor performance comes from degenerated components which drifted to the wrong hemisphere. We are confident that this problem can solved by forcing directions to the correct hemisphere in the EM algorithm.

Comparing the polynomial bases we can get some novel insights. First, using the full spherical SH basis without any modifications gives results similar to the grid based methods. Using specialized hemispherical bases produces better results in almost all cases. From those, Makhotkin's basis is worse than all others even in its orthogonalized form, which is much better than the original as shown in the supplemental 1. All other models (HSH, Zernike and generalized $\mathcal{H}$) have very similar error values and show better compression rations than grid based methods. From those three, HSH is often slightly worse than the other two. However, since the generalized $\mathcal{H}$ basis is not orthogonal, choices should consider all three models: HSH in case the target function has a constant value on horizon and $\mathcal{H}$ or Zernike otherwise.

6.3. Runtimes

The theoretical runtimes were already given at the end of the individual sections. They can be summarized in five categories:

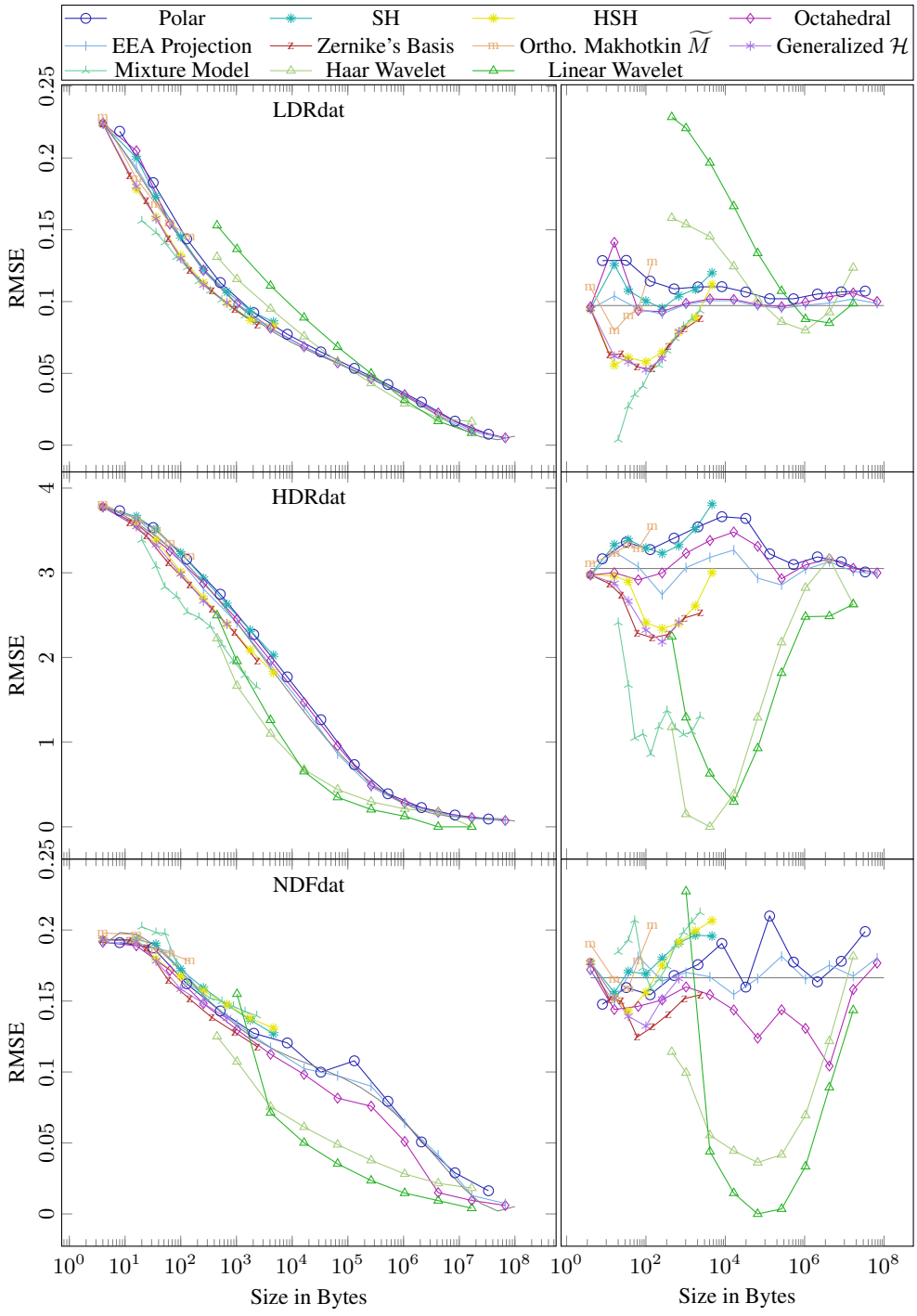


Figure 24. Error over size comparison for hemispherical models. On the right we subtracted offset polynomials to emphasis the differences between models. The polynomials are fitted to each data set individually and are visualized by the (straight) gray line.

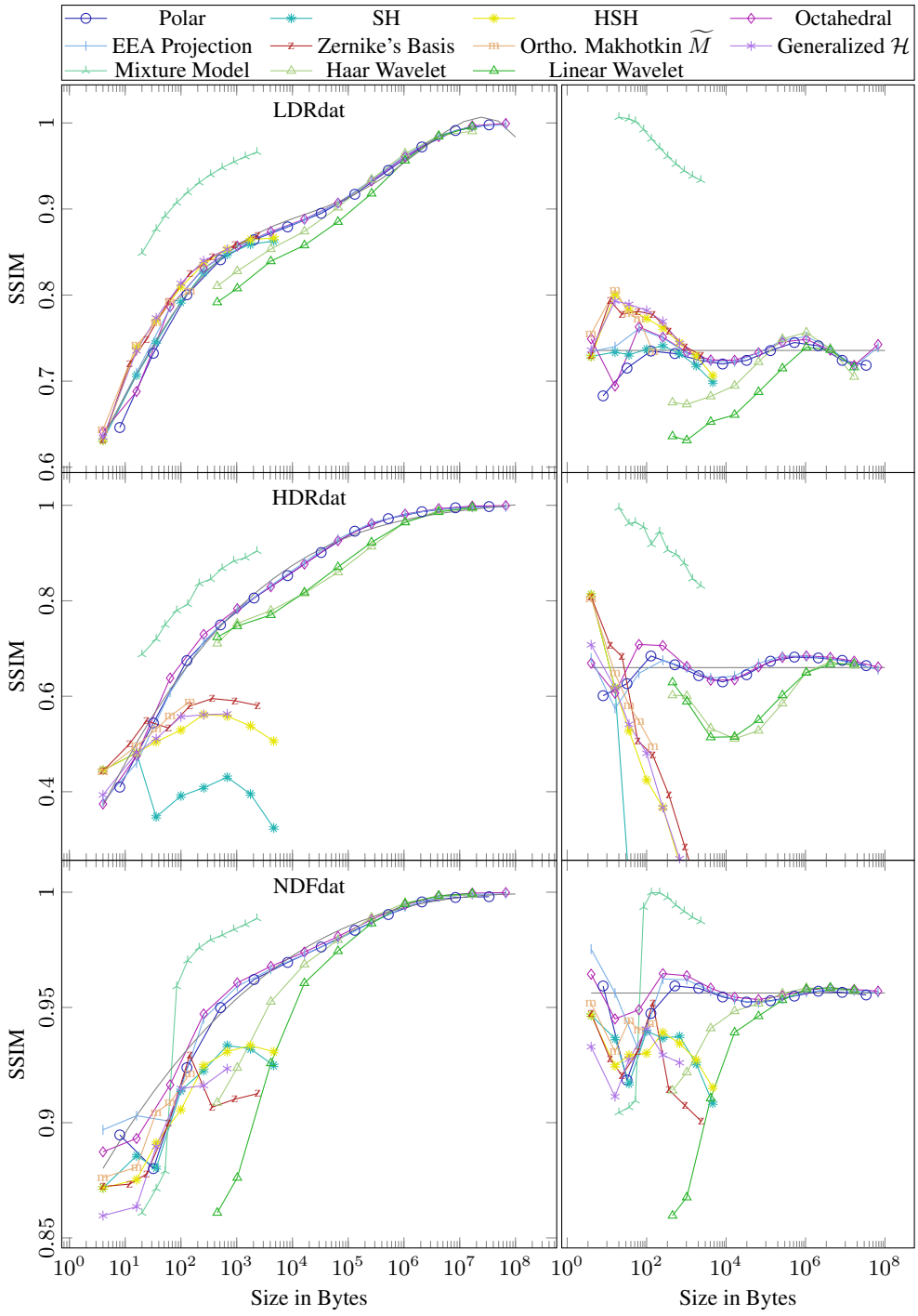


Figure 25. Structured similarity over size comparison for hemispherical models. On the right we subtracted offset polynomials to emphasis the differences between models. The polynomials are fitted to each data set and are visualized by the (straight) gray line.

	Fitting	Sampling
All mapping based models with n pixels	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Polynomial bases with l bands	$\mathcal{O}(l^2)$	$\mathcal{O}(l^2)$
Wavelet tree with f faces and k^2 pixels per face	$\mathcal{O}(n = fk^2)$	$\mathcal{O}(f \log k)$
n-component mixture models, weights only	$\mathcal{O}(n^3)$	$\mathcal{O}(n)$
n-component mixture models ($s > n$ input samples)	$\mathcal{O}(s^3)$	$\mathcal{O}(n)$

The fastest models are those based on a mapping. They can be fitted by resampling the target function with a constant cost per resulting pixel. A lookup is independent of the resolution except for cache reasons. The second fastest model is the wavelet transformation for which fitting has the same time complexity as for mapping based models and a lookup is possible in logarithmic time. Polynomial bases and mixture models are slowest and have a comparable lookup costs. In both cases it is determined by the number of basis functions/components. However, using tricks like k-nearest neighbor searches the lookup cost for sparse mixture models can be reduced. In any case is the projection to polynomial bases faster than the fitting of mixture models.

7. Extensions to the Bidirectional Domain

In computer graphics and physics there are bidirectional functions whose values depend on two independent directions. Examples are scattering functions on surfaces (BRDFs) and in free space (BSDFs). In a function $\mathbb{S}^2 \times \mathbb{S}^2 \mapsto \mathbb{X}$ each point of one sphere surface is another spherical function.

7.1. Parametrizations

Using two directions, there are two different choices for the parametrization. The first trivially uses the polar coordinates of both direction vectors. The second one represents the vectors relative to the average (half) vector. It is especially useful for BRDFs for reasons of feature alignment [Rusinkiewicz 1998]. Both are shown in Figure 26

In the first step the half vector of $\vec{x}_1$ and $\vec{x}_2$ is calculated as

$$\vec{h} = \frac{\vec{x}_1 + \vec{x}_2}{\|\vec{x}_1 + \vec{x}_2\|}.$$

Then two rotations are be applied to one of the vectors:

$$\vec{d} = \text{rotY}(-\theta_h) \cdot \text{rotZ}(-\varphi_h) \cdot \vec{x}_1.$$

The two rotations effectively align the half vector $\vec{h}$ with the z-axis. Note that the difference vector is always in the positive half space of the half-vector. It therefore has an angle $\theta_d \in [0, \pi/2]$. Also, it is possible to use a single rotation around $\vec{h} \times z$ which yields a different but similar parametrization.

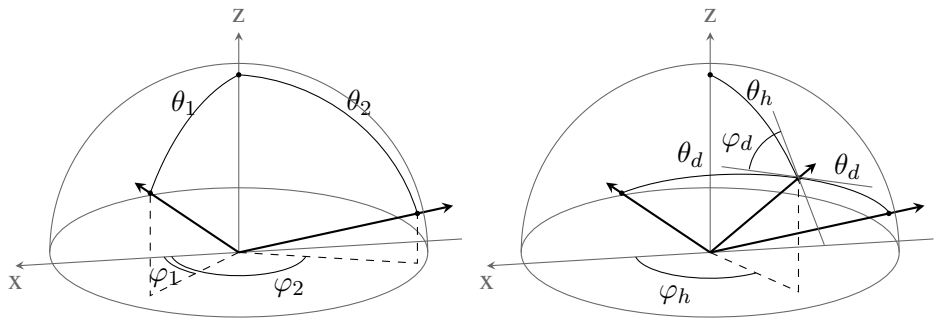


Figure 26. Two variants for the parametrization of bidirectional functions.

To revert the parametrization $\vec{x}_1$ can be computed by the inverse rotation and $\vec{x}_2$ by a reflection at $\vec{h}$ namely $\vec{x}_2 = 2(\vec{x}_1 \cdot \vec{h})\vec{h} - \vec{x}_1$. A fast implementation which avoids the heavy use of trigonometric functions is shown in listing 11.

```
# Map two directions onto half and difference vector.
reparamBidirToHalfDiff(dir0, dir1) -> (h, d)
    h = normalize(dir0 + dir1)
    # Construct rotation matrix rotY(-t)rotZ(-p) without trigonometry:
    sT = sqrt(1 - h.z*h.z)        # sin(theta) via trigonometric Pythagoras
    sP = sT < eps ? 0 : h.y / sT  # sin(phi)
    cP = sT < eps ? 1 : h.x / sT  # cos(phi)
    rYZ = ( h.z * cP, h.z * sP, -sT,
           -sP,       cP,       0,
            h.x,      h.y,      h.z)
    d = rYZ * dir0
end

# Revert the mapping.
reparamHalfDiffToBidir(h, d) -> (dir0, dir1)
    # Construct inverse rotation matrix transpose(rotY(-t)rotZ(-p)):
    sT = sqrt(1 - h.z*h.z)        # sin(theta) via trigonometric Pythagoras
    sP = sT < eps ? 0 : h.y / sT  # sin(phi)
    cP = sT < eps ? 1 : h.x / sT  # cos(phi)
    rYZinv = ( h.z * cP, -sP, h.x,
               h.z * sP,  cP, h.y,
              -sT,        0,  h.z)
    dir0 = rYZinv * d
    dir1 = 2 * dot(dir0, h) * h - dir0
end
```

Listing 11. Fast parametrization changes for bidirectional functions. The `eps` is to avoid divisions by 0 in practice.

7.2. Mapping Based Models

Many of the above models can be generalized to the twofold case. Each of the grid based mapping methods can be combined with each other by storing another map per cell. Thus, the two dimensional mapping becomes a four dimensional one. In some

cases, like isotropic reflection functions, one of the angles can be discarded. Then, a one-dimensional mapping of the remaining angle is the most appropriate choice.

7.3. Polynomial Bases

Spherical polynomial bases

$$f(\vec{d}) = \sum_i c_i \cdot b_i(\vec{d}).$$

have constant factors c_i which can be generalized into functions of a second direction:

$$f(\vec{d}_a, \vec{d}_b) = \sum_i c_i(\vec{d}_b) \cdot b_i(\vec{d}_a).$$

Now $c_i(\vec{d}_b)$ can be represented with the same or another basis recursively:

$$\begin{aligned} f(\vec{d}_a, \vec{d}_b) &= \sum_i \left(\sum_j c_{ij} \cdot b_j(\vec{d}_b) \right) \cdot b_i(\vec{d}_a) \\ &= \sum_i \sum_j b_i(\vec{d}_a) \cdot c_{ij} \cdot b_j(\vec{d}_b) \\ &= Y^T(\vec{d}_a) \cdot \mathbf{C} \cdot Y(\vec{d}_b). \end{aligned}$$

The reordering in the second line is possible due to commutativity and distributivity of sum and product. The third line is in matrix notation assuming that the bases b_i and b_j are the same. However, this is not necessary in general. It is also possible to use an entirely different basis or just another number of basis functions, if the two directions are of different semantic or importance.

If reciprocity $f(\vec{d}_a, \vec{d}_b) = f(\vec{d}_b, \vec{d}_a)$ is desired, which is the case for scattering functions, the coefficient matrix $\mathbf{C}$ must be symmetric. Further memory savings are possible if even more symmetries can occur with respect to the chosen basis. An example is the BRDF representation of [Westin et al. 1992] in SH-basis.

8. Conclusions

There are many different parametrizations, polynomial bases and mixture models described and tested above. The following list should give a guide on when to use which of the models. We discarded some of them in this list, because they either perform worse than others or are more complicated to implement. The winners are:

Projections (Section 3.3) The projections, we have shown, map hemispheres to a quadratic textures. They can be extended to spheres using two faces. For the mapping from disc to square two forms are available: Shirley's mapping and the Elliptical mapping which both perform similar with respect to RMSE.

- Very fast (mapping based)
- Only one texture for hemispherical functions

Cube/Isocube (Sections 3.2 and 3.7) Cube maps are among the most used maps and a lot of data can be found in this format. The extension to equal area pixel Isocubes can be added easily, but is not worth the extra costs for most scenarios.

- Very fast (mapping based)
- Hardware sampling support
- File formats like .dds and .ktx

HEALPix (Section 3.6) The HEALPix subdivision is the most regular grid we know of. If precision is of very high importance this map should be considered.

- Very fast (mapping based)
- Highly regular

Spherical Harmonics (Section 4.2) SHs are very smoothing representations which allow some nice mathematical tricks. Other than that, they are not better than mapping based approaches. To represent more details or to get a faster sampling prefer projections and cube maps. More bands lead to high computation times, ringing and numerical issues.

- Orthogonal base
- Rotation invariant

Mixture Models (Section 5) The best choice for small memory budgets, but hard to fit. The best fitting method is EM (Section 5.5) or reduction (Section 5.6) if a meaningful high resolution mixture is provided. Non-linear optimizations tend to local optimal more often.

- Highest compression in low ranges
- High adaption to HDR data
- Rotation invariant

Wavelets (Section 3.8) For bigger data the adaptive quad tree performs better than many other models. Therewith, Haar wavelets are much easier to implement, because they do not need neighborhoods, and should be preferred in subdivided spheres. Moreover, the wavelet compression can be done on 2D images like Projections or Cube maps.

- Highest compression in the mid to high ranges
- High adaption to edges
- Moderately fast

8.1. Open Issues

As you might expect there is not much to do considering spherical functions. The only really open problem is to find the globally optimal fit for mixture models. Further, wavelet transformations on the sphere seem to have space for improvements.

With respect to this document there are at least two open topics:

Principal Component Analysis In PCA a matrix is decomposed into eigenvectors and eigenvalues. Then, eigenvectors with a small contribution (small eigenvalue) are discarded for lossy compression / simplification of the data. It is possible to apply a PCA on all but the adaptive spherical mappings (i.e. all except wavelets and mixture models). It is therefore no individual mapping and not part of this work.

Spherical Harmonic Wavelets There are more approaches to the wavelet transformation on the sphere than using a tessellation. For example in [Lira et al. 2015] the SH wavelets are based on Legendre polynomials and form non-orthogonal bases on the sphere.

Acknowledgements

Coming soon.

References

AGARWAL, S., MIERLE, K., AND OTHERS, 2010. Ceres Solver. Accessed: 2016-12-12. URL: http://ceres-solver.org. 36

ARDESHIRI, T., ORGUNER, U., AND ÖZKAN, E. 2015. Gaussian Mixture Reduction Using Reverse Kullback-Leibler Divergence. *arXiv preprint arXiv:1508.05514*. URL: https://arxiv.org/abs/1508.05514. 2, 41

ASHIKHMIN, M., AND SHIRLEY, P. 2000. An Anisotropic Phong BRDF Model. *Journal of Graphics Tools (JGT) 5*, 25–32. URL: http://dx.doi.org/10.1080/10867651.2000.10487522. 33

BANERJEE, A., DHILLON, I. S., GHOSH, J., AND SRA, S. 2005. Clustering on the Unit Hypersphere using von Mises-Fisher Distributions. *Journal of Machine Learning Research*, 1345–1382. URL: http://jmlr.csail.mit.edu/papers/v6/banerjee05a.html. 38, 40

BECKMANN, P., AND SPIZZICHINO, A. 1987. The Scattering of Electromagnetic Waves from Rough Surfaces. *Norwood, MA, Artech House, Inc., 1987, 511 p.*. URL: http://adsabs.harvard.edu/abs/1987ah...book.....B. 31

BILMES, J. A., AND OTHERS. 1998. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. *International Computer Science Institute 4*, 510, 126. 38, 39

BUCHTA, C., KOBER, M., FEINERER, I., AND HORNIK, K. 2012. Spherical k-Means Clustering. *Journal of Statistical Software 50*, 10, 1–22. URL: http://www.jstatsoft.org/v50/i10/paper. 39

CASTAÑO, I., 2012. Seamless Cube Map Filtering. URL: http://the-witness.net/news/2012/02/seamless-cube-map-filtering/. 8

CIGOLLE, Z. H., DONOW, S., EVANGELAKOS, D., MARA, M., MCGUIRE, M., AND MEYER, Q. 2014. A Survey of Efficient Representations for Independent Unit Vectors. *Journal of Computer Graphics Techniques (JCGT) 3*, 2, 1–30. URL: http://jcgt.org/published/0003/02/01/. 2, 9, 10

CRASSIN, C., NEYRET, F., SAINZ, M., GREEN, S., AND EISEMANN, E. 2011. Interactive Indirect Illumination Using Voxel Cone Tracing. *Computer Graphics Forum (CGF) 30*, 7, 1921–1930. URL: https://research.nvidia.com/publication/interactive-indirect-illumination-using-voxel-cone-tracing. 2

CROUSE, D. F., WILLETT, P., PATTIPATI, K., AND SVENSSON, L. 2011. A Look at Gaussian Mixture Reduction Algorithms. In *Proc. of Conference on Information Fusion*, 1–8. URL: http://ieeexplore.ieee.org/document/5977695. 2, 41, 42

DEBEVEC, P. Light Probe Image Gallery. Accessed: 2015-02-05. URL: http://www.pauldebevec.com/Probes/. 4

DEMPSTER, A. P., LAIRD, N. M., AND RUBIN, D. B. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 1–38. URL: https://www.jstor.org/stable/2984875. 2, 3, 38

ENGELHARDT, T., AND DACHSBACHER, C. 2008. Octahedron Environment Maps. In *Proc. of Vision, Modeling, and Visualization (VMV)*, VMV, 383–388. URL: www.vis.uni-stuttgart.de/~engelhts/paper/vmvOctaMaps.pdf. 10

FISHER, R. 1953. Dispersion on a Sphere. *Proc. of Royal Society of London A: Mathematical, Physical and Engineering Sciences 217*, 1130, 295–305. URL: http://rspa.royalsocietypublishing.org/content/217/1130/295. 32

FREEDEN, W., SCHREINER, M., AND FRANKE, R. 1997. A Survey on Spherical Spline Approximation. *Surveys Mathematics for Industry 7*, 29–85. URL: http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:hbz:386-kluedo-5590. 32

GAUTRON, P., KRIVANEK, J., PATTANAIK, S. N., AND BOUATOUCH, K. 2004. A Novel Hemispherical Basis for Accurate and Efficient Rendering. In *Rendering Techniques '04 (Proc. EGSR)*, Eurographics Association, EGSR, 321–330. URL: https://diglib.eg.org/handle/10.2312/EGWR.EGSR04.321-330. 24

GAUVAIN, J.-L., AND LEE, C.-H. 1994. Maximum A Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains. *IEEE Transactions on Speech and Audio Processing 2*, 2, 291–298. URL: https://doi.org/10.1109/89.279278. 41

GORSKI, K. M., HIVON, E., BANDAY, A., WANDELT, B. D., HANSEN, F. K., REINECKE, M., AND BARTELMANN, M. 2005. HEALPix: a Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere. *The Astrophysical Journal 622*, 2, 759. URL: http://stacks.iop.org/0004-637X/622/i=2/a=759. 14

GREEN, R. 2003. Spherical Harmonic Lighting: The Gritty Details. In *Archives of the Game Developers Conference*, vol. 2, 2–3. URL: http://www.gdcvault.com/play/1022720/Spherical-Harmonic-Lighting-The-Gritty. 22

GREENE, N. 1986. Environment Mapping and Other Applications of World Projections. *IEEE Computer Graphics Applications 6*, 11, 21–29. URL: http://dx.doi.org/10.1109/MCG.1986.276658. 2

GREGER, G., SHIRLEY, P., HUBBARD, P. M., AND GREENBERG, D. P. 1998. The Irradiance Volume. *IEEE Computer Graphics and Applications 18*, 2, 32–43. URL: http://dx.doi.org/10.1109/38.656788. 2

GROSSMANN, A., AND MORLET, J. 1984. Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape. *SIAM Journal on Mathematical Analysis 15*, 4, 723–736. URL: http://dx.doi.org/10.1137/0515056. 17

HABEL, R., AND WIMMER, M. 2010. Efficient Irradiance Normal Mapping. In *Proc. of Symposium on Interactive 3D Graphics and Games*, I3D, 189–195. URL: http://www.cg.tuwien.ac.at/research/publications/2010/Habel-2010-EIN/. 3, 25

HEIDRICH, W., AND SEIDEL, H.-P. 1998. View-independent Environment Maps. In *Proc. of ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware (HWWS)*, ACM, HWWS '98, 39–ff. URL: http://doi.acm.org/10.1145/285305.285310. 9

HEITZ, E., DUPUY, J., CRASSIN, C., AND DACHSBACHER, C. 2015. The SGGX Microflake Distribution. *ACM Transactions on Graphics (TOG) 34*, 4, 48:1–48:11. URL: http://doi.acm.org/10.1145/2766988. 2

HORNIK, K., AND GRÜN, B. 2014. movMF: An R Package for Fitting Mixtures of von Mises-Fisher Distributions. *Journal of Statistical Software 58*, 1, 1–31. URL: https://www.jstatsoft.org/index.php/jss/article/view/v058i10. 38

ISIDORO, J. R., AND MITCHELL, J. L. 2005. Angular Extent Filtering with Edge Fixup for Seamless Cubemap Filtering. In *ACM SIGGRAPH Sketches*, ACM, 33. URL: http://doi.acm.org/10.1145/1187112.1187151. 8

JAKOB, W., REGG, C., AND JAROSZ, W. 2011. Progressive Expectation–Maximization for Hierarchical Volumetric Photon Mapping. *Computer Graphics Forum (Proc. EGSR) 30*, 4 (June). URL: http://dx.doi.org/10.1111/j.1467-8659.2011.01988.x. 41

JENDERSIE, J., KURI, D., AND GROSCH, T. 2016. Real-Time Global Illumination Using Precomputed Illuminance Composition with Chrominance Compression. *Journal of Computer Graphics Techniques (JCGT) 5*, 4 (Dec.), 8–35. URL: http://jcgt.org/published/0005/04/02/. 22

KENT, J. T. 1982. The Fisher-Bingham Distribution on the Sphere. *Journal of the Royal Statistical Society. Series B (Methodological) 44*, 1, 71–80. URL: https://www.jstor.org/stable/2984712. 33

KENT, J. T. 1994. The Complex Bingham Distribution and Shape Analysis. *Journal of the Royal Statistical Society. Series B (Methodological) 56*, 2, 285–299. URL: http://www.jstor.org/stable/2345900. 33, 41

KENT, J. T. 1997. Data Analysis for Shapes and Images. *Journal of Statistical Planning and Inference 57*, 2, 181 – 193. URL: http://dx.doi.org/10.1016/S0378-3758(96)00043-2. 33, 41

KOENDERINK, J. J., VAN DOORN, A. J., AND STAVRIDI, M. 1996. Bidirectional Reflection Distribution Function Expressed in Terms of Surface Scattering Modes. In *Proc. of European Conference on Computer Vision*, 28–39. URL: http://dx.doi.org/10.1007/3-540-61123-1_125. 26, 27

KRIVANEK, J., GAUTRON, P., PATTANAIK, S., AND BOUATOUCH, K. 2005. Radiance Caching for Efficient Global Illumination Computation. *IEEE Transactions on Visualization and Computer Graphics 11*, 5, 550–561. URL: https://doi.org/10.1109/TVCG.2005.83. 2, 22

KURT, M., AND EDWARDS, D. 2009. A Survey of BRDF Models for Computer Graphics. *ACM SIGGRAPH Computer Graphics 43*, 2, 4:1–4:7. URL: http://doi.acm.org/10.1145/1629216.1629222. 2

LALONDE, P. 1997. A Wavelet Representation of the BRDF. *IEEE Transactions on Visualization and Computer Graphics 3*, 329–336. URL: http://dx.doi.org/10.1109/2945.646236. 18

LEUNG, C.-S., WONG, T.-T., LAM, P.-M., AND CHOY, K.-H. 2006. An RBF-based Compression Method for Image-Based Relighting. *IEEE Transactions on Image Processing (TIP) 15*, 4, 1031–1041. URL: https://doi.org/10.1109/TIP.2005.863936. 35

LEVENBERG, K. 1944. A Method for the Solution of Certain Problems in Least Squares. *Quarterly of Applied Mathematics 2*, 164–168. URL: https://www.jstor.org/stable/43633451. 2, 36

LIRA, M. M., DE OLIVEIRA, H. M., CARVALHO JR, M., AND DE SOUZA, R. 2015. Compactly Supported Wavelets Derived from Legendre Polynomials: Spherical Harmonic Wavelets. *arXiv preprint arXiv:1502.00950*. URL: https://arxiv.org/abs/1502.00950. 55

LOUNSBERY, M., DEROSE, T. D., AND WARREN, J. 1997. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. *ACM Transactions on Graphics (TOG) 16*, 1, 34–73. URL: http://doi.acm.org/10.1145/237748.237750. 18

MAKHOTKIN, O. A. 1996. Analysis of Radiative Transfer Between Surfaces by Hemispherical Harmonics. *Journal of Quantitative Spectroscopy and Radiative Transfer 56*, 6, 869–879. URL: https://doi.org/10.1016/S0022-4073(96)00040-4. 3, 28, 29

MARDIA, K. V., AND JUPP, P. E. 2000. *Directional Statistics*. John Wiley & Sons. URL: http://onlinelibrary.wiley.com/book/10.1002/9780470316979. 41

MARQUARDT, D. W. 1963. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal on Applied Mathematics 11*, 2, 431–441. URL: https://www.jstor.org/stable/1267303. 2, 36

M.I.A. LOURAKIS, 2004. levmar: Levenberg-Marquardt nonlinear least squares algorithms in {C}/{C}++. Accessed: 2016-12-12. URL: http://users.ics.forth.gr/~lourakis/levmar/. 36

MOLLWOLLFUMBLE, 2011. Subrandom Numbers. Accessed: 2017-05-10. URL: http://mollwollfumble.blogspot.de/. 35

MONTES, R., AND UREÑA, C. 2012. An Overview of BRDF Models. Technical 2012-001. URL: http://digibug.ugr.es/bitstream/10481/19751/1/rmontes_LSI-2012-001TR.pdf. 2

NARCOWICH, F. J., AND WARD, J. D. 1996. Nonstationary Wavelets on the m-Sphere for Scattered Data. *Applied and Computational Harmonic Analysis 3*, 4, 324–336. URL: www.math.tamu.edu/~fnarc/psfiles/sbf_rev.ps. 32, 34

NEAL, R. M., AND HINTON, G. E. 1998. A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants. In *Learning in Graphical Models*. MIT Press, 355–368. URL: http://dx.doi.org/10.1007/978-94-011-5014-9_12. 41

NYQUIST, H. 1928. Certain Topics in Telegraph Transmission Theory. *Transactions of the American Institute of Electrical Engineers 47*, 2, 617–644. URL: https://doi.org/10.1109/T-AIEE.1928.5055024. 21

PERSSON, E. Humus' Cube Map Collection. Accessed: 2015-02-05, License: CC BY 3.0. URL: http://www.humus.name/index.php?page=Textures. 4

RUNNALLS, A. R. 2007. Kullback-Leibler Approach to Gaussian Mixture Reduction. *IEEE Transactions on Aerospace and Electronic Systems 43*, 3, 989–999. URL: https://doi.org/10.1109/TAES.2007.4383588. 2, 3, 42

RUSINKIEWICZ, S. M., 1997. A Survey of BRDF Representation for Computer Graphics. URL: http://www.cs.princeton.edu/smr/cs348c-97/surveypaper.html. 2

RUSINKIEWICZ, S. M. 1998. A New Change of Variables for Efficient BRDF Representation. In *Rendering Techniques' 98 (Proc. EGWR)*, 11–22. URL: http://dx.doi.org/10.1007/978-3-7091-6453-2_2. 51

SCHRÖDER, P., AND SWELDENS, W. 1995. Spherical Wavelets: Efficiently Representing Functions on the Sphere. In *Proceedings of SIGGRAPH '95*, SIGGRAPH, 161–172. 18, 19

SHANNON, C. E. 1948. A Mathematical Theory of Communication. *Bell System Technical Journal 27*, 4, 623–656. URL: http://ieeexplore.ieee.org/document/6773067/. 21

SHIRLEY, P., AND CHIU, K. 1997. A Low Distortion Map Between Disk and Square. *Journal of Graphics Tools (JGT) 2*, 3, 45–52. URL: http://dx.doi.org/10.1080/10867651.1997.10487479. 9, 15

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed Radiance Transfer for Real-time Rendering in Dynamic, Low-frequency Lighting Environments. *ACM Transactions on Graphics (TOG) 21*, 3, 527–536. URL: http://doi.acm.org/10.1145/566654.566612. 2, 22

SLOAN, P.-P. 2008. Stupid Spherical Harmonics (SH) Tricks. In *Archives of the Game Developers Conference*, vol. 9. URL: http://www.gdcvault.com/play/273/Stupid-Spherical-Harmonics-(SH). 21, 22

SLOAN, P.-P. 2013. Efficient Spherical Harmonic Evaluation. *Journal of Computer Graphics Techniques (JCGT) 2*, 2, 84–90. URL: http://jcgt.org/published/0002/02/06/. 23

SWELDENS, W. 1998. The Lifting Scheme: A Construction of Second Generation Wavelets. *SIAM Journal on Mathematical Analysis 29*, 2, 511–546. URL: http://dx.doi.org/10.1137/S0036141095289051. 18

TARINI, M., HORMANN, K., CIGNONI, P., AND MONTANI, C. 2004. PolyCube-Maps. *ACM Transaction on Graphics (TOG) 23*, 3, 853–860. URL: http://doi.acm.org/10.1145/1015706.1015810. 8

TSAI, Y.-T., AND SHIH, Z.-C. 2006. All-Frequency Precomputed Radiance Transfer Using Spherical Radial Basis Functions and Clustered Tensor Approximation. *ACM Transactions on Graphics (TOG) 25*, 967–976. URL: http://doi.acm.org/10.1145/1141911.1141981. 34

VARDIS, K., PAPAIOANNOU, G., AND GKARAVELIS, A. 2014. Real-Time Radiance Caching using Chrominance Compression. *Journal of Computer Graphics Techniques (JCGT) 3*, 4, 111–131. URL: http://jcgt.org/published/0003/04/06/. 2, 22

VORBA, J., KARLÍK, O., ŠIK, M., RITSCHEL, T., AND KŘIVÁNEK, J. 2014. On-line Learning of Parametric Mixture Models for Light Transport Simulation. *ACM Transactions on Graphics (TOG) 33*, 4. URL: http://cgg.mff.cuni.cz/~jaroslav/papers/2014-onlineis/. 38

WALTER, B., MARSCHNER, S. R., LI, H., AND TORRANCE, K. E. 2007. Microfacet Models for Refraction Through Rough Surfaces. In *Proc. of Eurographics Symposium on Rendering (EGSR)*, Eurographics Association, EGSR'07, 195–206. URL: http://dx.doi.org/10.2312/EGWR/EGSR07/195-206. 31

WAN, L., WONG, T.-T., AND LEUNG, C.-S. 2005. Spherical Q2-tree for Sampling Dynamic Environment Sequences. In *Proc. of Eurographics Symposium on Rendering (EGSR)*, Eurographics Association, EGSR, 21–30. URL: http://dx.doi.org/10.2312/EGWR/EGSR05/021-030. 14

WAN, L., WONG, T.-T., AND LEUNG, C.-S. 2007. Isocube: Exploiting the Cubemap Hardware. *IEEE Transactions on Visualization and Computer Graphics 13*, 4, 720–731. URL: https://doi.org/10.1109/TVCG.2007.1020. 15

WANG, Z., BOVIK, A. C., SHEIKH, H. R., AND SIMONCELLI, E. P. 2004. Image Quality Assessment: from Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing (TIP) 13*, 4, 600–612. URL: https://doi.org/10.1109/TIP.2003.819861. 5

WARD, G. J. 1992. Measuring and Modeling Anisotropic Reflection. *Computer Graphics (Proc. SIGGRAPH) 26*, 2, 265–272. URL: http://doi.acm.org/10.1145/142920.134078. 33

WEST, M. 1993. Approximating Posterior Distributions by Mixture. *Journal of the Royal Statistical Society. Series B (Methodological)*, 409–422. URL: http://dx.doi.org/10.1063/1.1835242. 42

WESTIN, S. H., ARVO, J. R., AND TORRANCE, K. E. 1992. Predicting Reflectance Functions from Complex Surfaces. *Computer Graphics (Proc. SIGGRAPH) 26*, 2, 255–264. URL: http://doi.acm.org/10.1145/142920.134075. 53

XU, K., SUN, W.-L., DONG, Z., ZHAO, D.-Y., WU, R.-D., AND HU, S.-M. 2013. Anisotropic Spherical Gaussians. *ACM Transactions on Graphics (TOG) 32*, 6, 209. URL: http://doi.acm.org/10.1145/2508363.2508386. 32, 34

ZAAL, G. HDRI Haven. Accessed: 2017-04-07, License: CC BY 4.0. URL: http://hdrihaven.com. 4

ZERNIKE, V. F. 1934. Beugungstheorie des Schneidenverfahrens und seiner Verbesserten Form, der Phasenkontrastmethode. *Physica 1*, 689–704. 26

Author Contact Information

Johannes Jendersie
TU-Clausthal
Germany
www.jojendersie.de