BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2019
http://excel.fit.vutbr.cz

# Importance sampling probability density functions represented with (hemi)spherical harmonics

Michal Vlnas*

**Abstract**

This paper discusses a novel approach for 3D importance sampling probability density functions represented as spherical harmonics (SH). The proposed method pre-samples all spherical harmonic basis functions using a pseudo-random number generator (PRNG) with known seed and decides which samples will be used for further sampling and records these samples. Any function then can be reconstructed given a vector of SH coefficients and PRNG. This approach has immediate usability in rendering, e. g. global illumination, radiance transfer, etc. In the contrary to existing methods, our approach does not require evaluation of any function integrals. The proposed approach generates over 3 million sample per seconds (while using single core) and does not decrease performance with increased size of SH basis.

**Keywords:** spherical harmonics — importance sampling — probability density function — pseudo-random number generators

**Supplementary Material:** *N/A*

*xvlnas00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

## 1. Introduction

Spherical harmonics (SH) are special functions defined on the sphere domain. They are important in many nature applications, such as electrostatic and electromagnetic fields, atomic orbital electron configurations, gravitational fields, or the magnetic fields of planetary bodies and stars.

In computer graphics, spherical harmonics can be also widely used in many different applications, since many problems are defined over the (hemi)spherical domain. An extensive research was focused in the area of precomputed radiance transfer by Kautz, Snyder et al. [1, 2]. Another popular technique is irradiance normal mapping, which is mostly used in games, where the irradiance signal can be efficiently represented with orthonormal basis functions, such as spherical harmonics. In Wimmer and Habel work [3] can be also seen, that not even a full spherical basis is needed, as irradiance can be accurately represented using only a few coefficients. Spherical harmonics also play a role in modeling of 3D shapes and it's description [4]. More intuitive view, focused on practical ways and techniques, especially in lightning simulations using spherical harmonics, can be also found in R. Green work [5].

Importance sampling (generating samples according to the intensity distribution of a function) is widely used in many disciplines, as it can lead to significantly better results. In rendering, Monte Carlo integrations are very common in situations where importance sampling techniques can be used to reduce variance. However, they have widespread usage in other branches, e. g. procedural geometry, etc.

## 2. Background

The first practical SH importance sampling was presented by Jarosz et al. [6], which is based on hierarchical sample warping according to the function integrals. Such approach requires a lot of effort for integral evaluation even though most of the values can be precomputed, it still suffers from performance drops when using a high number of basis functions. Comparing to their method, our approach does not need to evaluate/-store any function integrals.

### 2.1 Spherical harmonics

Spherical harmonics (also called as Laplace spherical harmonics, shortened to SH) represent spherical functions, as they are completely orthogonal functions on a sphere. They are based on associated Legendre polynomials $P_l^m(x)|_{x=\cos\theta}$, which can be defines using recurrence relations [1], see Equation 1.

$$\begin{aligned} P_m^m(x) &= (-1)^m(2m-1)!!(1-x^2)^{m/2} \\ P_{m+1}^m(x) &= x(2m+1)P_m^m \\ (l-m)P_l^m(x) &= x(2l-1)P_{l-1}^m - (l+m-1)P_{l-2}^m \end{aligned} \tag{1}$$

If a direction $\vec{\omega}$ is represented using standard spherical parameterization, $\vec{\omega} = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$, then the real-valued spherical harmonic basis functions are defined in Equation 2.

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

where $P_l^m$ are the ALPs mentioned above and $K_l^m$ are normalization constants defined as:

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

The constants $l$ and $m$ are integer values which represent the order $l$ and the degree $m$ of the basis functions. Formally, $l$ is required to be a non-negative number and $m$ should satisfy the condition $-l \leq m \leq l$.

Spherical harmonics are also orthonormal and rotation invariant [7]. Demonstration of SH is shown in Figure 1.

#### 2.1.1 Approximation

Any real-valued spherical function can be approximated (see Equation 3) with SH by using a linear combination of spherical harmonics [5] basis functions:

$$f(\theta,\phi) \approx \sum_{l=0}^{N-1}\sum_{m=-l}^{l} f_l^m \cdot y_l^m(\theta,\phi) \tag{3}$$

where $f_l^m$ are function coefficients computed by projecting a function onto each SH basis. These values can be estimated using various approaches, including Monte Carlo estimations.

### 2.2 Hemispherical harmonics

As SH represents the whole spherical domain, in some cases, it might be useful to limit domain only to hemispherical (see Figure 2). The first set of hemispherical basis functions was introduced by Gautron et al. [8].

[1] operator !! stands for double factorial

These functions are still based on associated Legendre polynomials with shifted domain (Equation 4) using a presented linear transformation.

$$\widetilde{P}_l^m(\cos\theta) = P_l^m(2\cos\theta - 1) \text{ with } \theta \in [0, \frac{\pi}{2}) \tag{4}$$

HSH are then orthogonal over $\langle 0, \frac{\pi}{2}\rangle \times \langle 0, 2\pi)$, so the corresponding normalization constant is then:

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

### 2.3 Monte Carlo techniques

Monte Carlo methods are a class of very common computational algorithms that rely on random sampling in order to provide estimated results. In the principle, any problem described with probabilistic model can be solved using Monte Carlo. Practically, Monte Carlo is mostly being used for numerical integration. In this case, it uses a kind of sampling from given probabilistic model to evaluate definite integrals. Many different approaches can be used, such as uniform, stratified or importance sampling. For all of these techniques, it is required to have a pseudo-random number generator and probabilistic description of function, e. g. spherical harmonics in our case.

### 2.4 Pseudo-random number generators

Pseudo-random number generators (PRNG) are completely deterministic algorithms for generating a sequence of numbers whose distribution is obviously not random, while it often can be acceptable by various algorithms, but that can be effectively determined by an initial value – seed. Sequence reproducibility is an important property for the proposed algorithm. Length of the period and uniform distribution of samples are crucial properties of these generators as well. Mersenne Twister [9] has one of the longest periods with proven equidistribution. But on the other hand, xor-shifts [10] are one of the fastest generators.

## 3. Proposed approach

As mentioned above, in Equation 3, any spherical function can be approximated using the weighted sum of (hemi)spherical harmonics. In the proposed algorithm, it is assumed that each basis function coefficient $f_l^m$ is a probability of picking the basis function so the coefficients must fit the condition in Equation 5.

$$\sum_{l=0}^{N-1}\sum_{m=-l}^{l} f_l^m = 1 \tag{5}$$

In the following text, we are assuming usage of hemispherical harmonics as in this case we can use

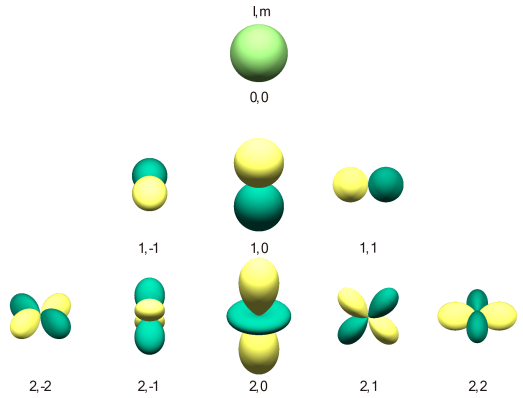



**Figure 1.** Spherical harmonic basis functions, up to degree $l = 2, m = 2$

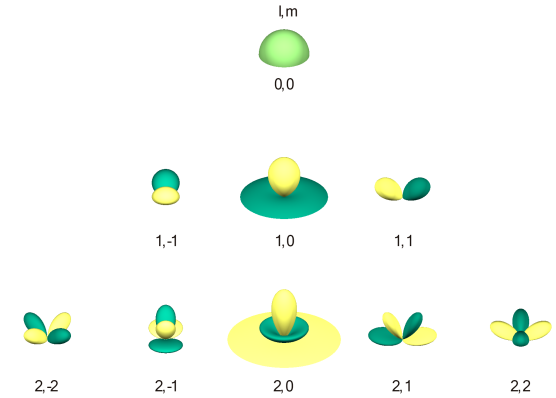



**Figure 2.** Hemispherical harmonics, up to degree $l = 2, m = 2$

**Algorithm 1:** PRE-SAMPLING AND FILTERING

**Data:** $N$ is total number of samples
**Data:** $y_l^m$ is basis function
**Data:** *seed* for random number generator
**Result:** $S$ is a skipping sequence

```
// pre-sample basis function
for i ← 0 to N − 1 do
    u, v ← random_float(seed)
    θ, φ ← sample_hemisphere(u, v)
    samples[i] = y_l^m(θ, φ)
end
// normalize values to [0, 1)
normalize(samples)
// initialize rejecting step
step = 0
foreach s ∈ samples do
    r_1 ← random_float(seed)
    // threshold rejecting
    if r_1 > s then
        step++
        continue
    end
    // store skipping interval
    S ← insert(step)
    // reset current "skip" step
    step = 0
end
// reset PRNG state to default
reset(seed)
return S
```

the approximated form as the probability density function of the distribution given by coefficients $f_l^m$, while using the whole spherical domain would break the probabilistic rules.

### 3.1 Pre-sample and filter

The crucial part of the proposed approach is preprocessing. The main idea behind this part is to importance sample each basis function independently (each function independently on the other ones).

As it can be seen in Algorithm 1, at first, each basis function is uniformly sampled (using projection to the hemisphere) from given PRNG seed. It is needed to normalize all the sampled function values to the interval $[0,1)$, as naturally they are out of this interval. Also, this normalization constant needs to be kept, as it is needed when assembling a function from SH basis functions.

The latter step is to filter these samples so that they are "importance distributed" according to the function value. Filtering is based on *threshold rejecting*, samples are compared to the random threshold and then rejected or accepted. This is the most important part of preprocessing, as when the sample is rejected, it is noted (recorded) and when one of the following generated samples is accepted, the algorithm stores information about the number of previously skipped samples so that it creates a sequence of these skipping intervals – an intuitive view can be seen in Figure 3 a), the skipping interval is meant to be the gap between the accepted samples. Finally, it is required to reset the state of PRNG. Otherwise it would not be possible to reproduce the selected samples during online sampling.

From the implementation point of view, according to our observations, skipping steps can be safely stored into the single bytes to efficiently optimize memory usage. When generating a large number of preprocessed samples, so that it would not fit in the random access memory, it is recommended to pre-load sequences from the storing device to not decelerate the performance of the sampling.

### 3.2 On-line sampling

Before the actual sampling, it is needed to assembly an approximated function. Each basis function requires a weights, from given set of weights (estimated with some kind of analytical or numerical method – depends

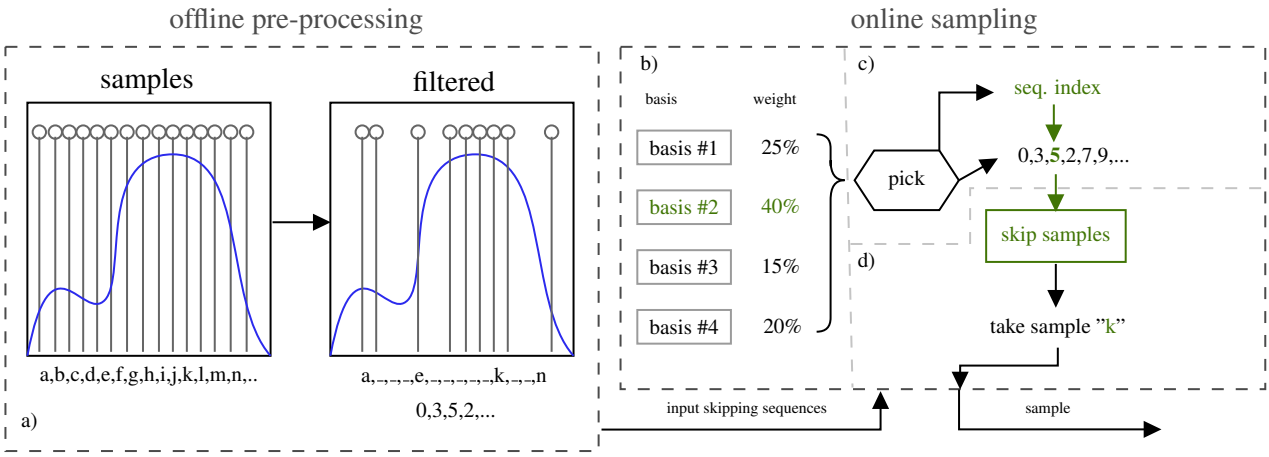



**Figure 3.** Algorithm illustration: a) generate $N$ random samples (for clarity a-n), filter out minor samples, using random threshold rejecting, so the probability distribution is preserved b) each basis function has assigned weight, pseudo-random generator with seed and skipping sequence c) pick basis according to the weight distribution, get it's skipping sequence, including current position inside, currently pointing on 5 d) skip $x$ samples and take the following sample "k"

**Algorithm 2:** ON-LINE SAMPLING

**Data:** $F[l(l+1)+m]$ is a vector of function coefficients
**Data:** $S$ is a pre-defined skipping sequence
**Data:** *seed* for random number generator
**Data:** $i$ is an index in the sequence
**Result:** $\omega$ is a sampled direction
**Result:** $p$ is a probability of sampling $\omega$

```
// select basis w.r.t weights in F
y_l^m, p ← pick_basis(F)
// two random numbers per sample
step ← 2 · S[i++]
// skipping
for k ← 0 to step do
    random_skip(seed)
end
// sample from filtered set
u, v ← random(seed)
θ, φ ← sample_hemisphere(u, v)
// update sampling probability
p = p * normalize(y_l^m(θ, φ))
return ω(θ, φ), p
```

on application). All these weights are multiplied with basis normalization constant and then re-normalized, so they sum into one. Practically, it arranges a balance between the given weights and the basis function norms. Function is then defined with $N$ weighted basis functions, what corresponds to Equation 3.

### Sampling

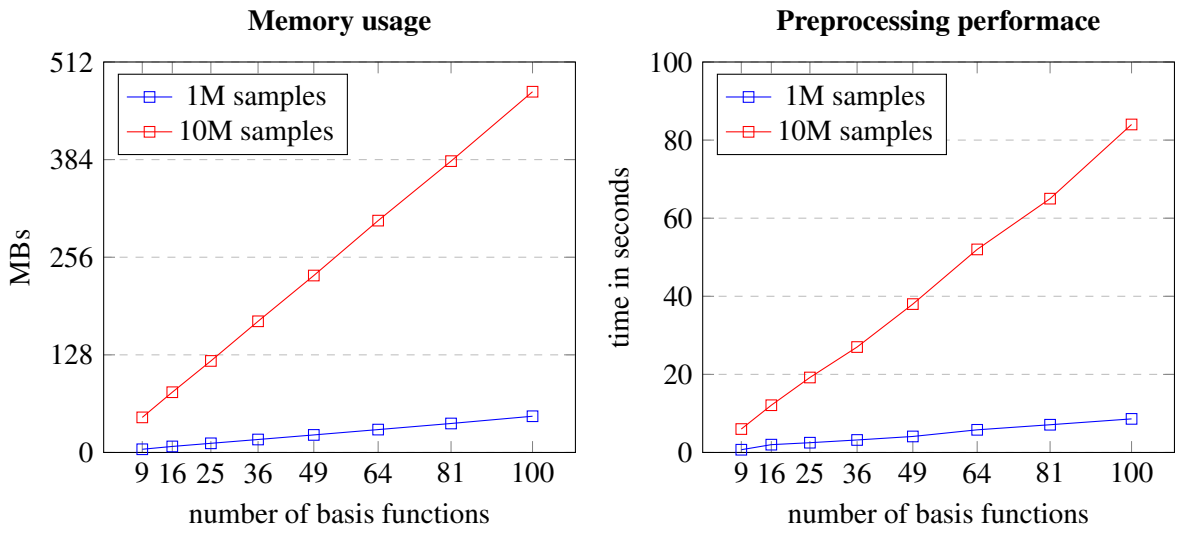
Given a skipping sequence, the function sampling is then straightforward using of Algorithm 2. At first, it is needed to choose a basis according to given vector of function coefficients – it corresponds to weighted pick from set, see Figure 3b. Afterwards, using the selected basis function, one can get a number of samples to be skipped (Figure 3c) – assuming that we have stored an index variable to indicate the position in the sequence. Practically, sample is a composited number from two-random numbers generated with the RNG, so the skipping is just switching the RNG state. After skipping the selected number of samples, the resulting sample is just the next generated sample with pseudo-random number generator, projected onto hemisphere, as the resulting sample is in spherical coordinates.

The probability of sampling a direction $\omega$ can be evaluated as product of basis weight (normalized to $[0,1)$ and function value in sampled direction, normalized according to basis norm.

## 4. Results

The fundamental question is about performance. Because of the implementation manner of the algorithm, sampling performance remains the same, independently on number of the basis functions. It is a remarkable advantage against other methods. The proposed method samples about 3 million samples per second while using the only one core. On the other hand, memory complexity is raising with increasing number of coefficients. The memory consumption is shown in Graph 4a using input of 1 to 10 millions of samples. However, the real memory consumption would be quite different, when using some kind of effective storage, as the samples are used only once, there is no need to keep them in the memory, they can be safely stored on the hard drive and synchronously loaded during sampling, so there would be just a constant number of samples in the memory.

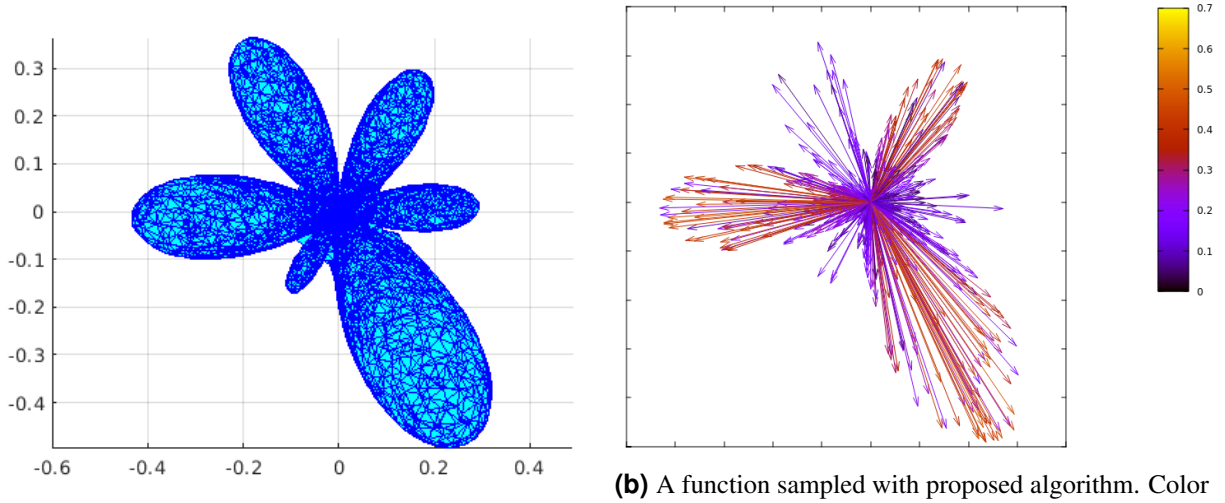
Also, preprocessing performance measured in real time (in seconds) can be seen in Graph 4b. Both graphs




**(a)** A graph showing memory consumption of proposed method

**(b)** Preprocessing stage performance dependent on grade and number of samples

**Figure 4.** Memory and performance graphs




**(a)** An example of function surface

**(b)** A function sampled with proposed algorithm. Color represents the probability of sampling a direction

**Figure 5.** Hemispherical function, both plots are top view for better clarity

are dependent on the number of basis functions. All the values were measured using Intel(R) Core(TM) i5-4460 CPU @ 3.20GHz processor.

A demonstration of algorithm is shown in Figure 5. An example of hemispherical function was generated, shown in Figure 5a, using 4 different hemispherical harmonics basis function up to grade 3. As was already mentioned, that the goal of this paper is to present a method for representing a probability density functions, so the sum of function coefficients is equal to 1. On the other side, in Figure 5b, one can see an importance sampled function, using the same set of basis functions and 500 samples.

No comparison to state of the art methods is presented, as the proposed method is quite different in it's approach and also the practical usage of the algorithm is not complete yet – making the actual comparison very complicated. For that reason, state of the art comparison is still an opened question which will be answered in the future.

## 5. Conclusions

In this paper, we presented a new technique for fast importance sampling of probability density functions represented with spherical harmonics. The main advantage of proposed algorithm is its constant performance which is independent on the count of given coefficients. The probability density function modeled using the spherical harmonics could be used as a black box in guided renderer, as most of the techniques nowadays lacks the simplicity and requires a lot of computational time to make them efficient.

The future work includes efficient memory management for high order sequences. As it has been

already said in previous section, one approach could be to pre-load enough numbers, required for on-line sampling, and discard the already used ones, as the generated numbers can be used only once. In addition to this, a question about how many samples to pre-generate is still open. In some situation it is possible to get notion about this, but mostly it is unknown. However, every time, it is possible to generate new data set of samples, not only offline, but also on the fly.

## Acknowledgements

I would like to thank my supervisor, Pavel Zemčík, for his ideas in introducing this method and substantial help.

## References

[1] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.*, 21(3):527–536, July 2002.

[2] Jan Kautz, John Snyder, and Peter-Pike J Sloan. Fast arbitrary brdf shading for low-frequency lighting using spherical harmonics. *Rendering Techniques*, 2(291-296):1, 2002.

[3] Ralf Habel and Michael Wimmer. Efficient irradiance normal mapping. pages 189–195, 02 2010.

[4] Mohamed Mousa, Raphaëlle Chaine, and Samir Akkouche. Direct spherical harmonic transform of a triangulated mesh. *Journal of graphics tools*, 11(2):17–26, 2006.

[5] Robin Green. Spherical harmonic lighting: The gritty details. In *Archives of the Game Developers Conference*, volume 56, page 4, 2003.

[6] Wojciech Jarosz, Nathan A. Carr, and Henrik Wann Jensen. Importance sampling spherical harmonics. *Computer Graphics Forum (Proceedings of Eurographics)*, 28(2):577–586, apr 2009.

[7] Cheol Ho Choi, Joseph Ivanic, Mark S Gordon, and Klaus Ruedenberg. Rapid and stable determination of rotation matrices between spherical harmonics by direct recursion. *The Journal of Chemical Physics*, 111(19):8825–8831, 1999.

[8] Pascal Gautron, Jaroslav Krivanek, Sumanta N. Pattanaik, and Kadi Bouatouch. A novel hemispherical basis for accurate and efficient rendering. *ACM SIGGRAPH 2007 Papers - International Conference on Computer Graphics and Interactive Techniques*, pages 321–330, 06 2004.

[9] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, January 1998.

[10] George Marsaglia. Xorshift rngs. *Journal of Statistical Software, Articles*, 8(14):1–6, 2003.