A Compression Method For Spectral Photon Map Rendering

Gorm Lai
Department of Computer Science,
University of Aarhus and
Deadline Games
Fabriksmestervej 4-6
Denmark, 1437 Copenhagen K
gorm.lai@deadlinegames.com

Niels Jørgen Christensen
Informatics and Mathematical Modelling,
Technical University of Denmark
Building 321
Denmark, 2800 Lyngby
njc@imm.dtu.dk

ABSTRACT

The photon map method can easily be extended to handle wavelength-dependent phenomena such as dispersion, chromatic aberration, etc. Using the computationally effective approach of point sampling for this extension, the size of the photon map is increased proportionally with the number of samples. In this paper we present a solution for modelling wavelength-dependent phenomena that keeps variance low, while having a memory usage comparable to that of an RGB based renderer. The method is best used for photon mapping, where there is a need to store large amounts of spectral flux directly in the photon map. Our method incurs a slight loss of accuracy for photons in the global map, while photons in the caustics map retain all information. Our tests show negligible loss of accuracy in the image quality.

Keywords: Global illumination, photon mapping, spectral power distribution, rendering, Monte Carlo methods, refraction, rendering equation

1 INTRODUCTION

In the sense of computer graphics, the discipline of global illumination has been around for over 20 years. The problem was first formalized in a ground breaking paper by [Kaj86], called *The Rendering Equation* and introduced a way of solving the equation, and introduced path tracing as a way of solving it. Other important advances in global illumination include radiosity [Gor84], irradiance caching [War88], bi-directional path tracing [Laf93] and photon mapping [Jen01]. The extension of the rendering equation to make it wavelength-dependent is straightforward, and is given in Equation 1, where λ represents the wavelength.

$$L(x,\vec{\omega},\lambda) = L_e(x,\vec{\omega},\lambda) + \int_\Omega f_r(x,\vec{\omega},\vec{\omega}',\lambda)L(x',\vec{\omega}',\lambda)(\vec{\omega}'\cdot\vec{n})d\vec{\omega}' \quad (1)$$

Many people have tried solving Equation 1 using the naive approach of solving it one discrete wavelength at a time. However, this method is time consuming and introduces variance.

[Col94] made a spectrally global illumination system, based on Backwards Ray Tracing [Arv86]. Collins uses sampling for the representation of his spectral power distributions. He uses 6 bins to represent a spectrum. One problem that Collins runs into is that the Illumination Map, which stores the particle power depositions, uses quite some memory.

[Wil00] extends a standard ray tracer to handle dispersion. Instead of handling dispersion in a deterministic fashion, where a spectral power distribution of N samples is split into N sub-spectra, a sub-spectrum is jittered by as much as half the width of the wavelength band represented by the specific bin. This approach eliminates some of the banding problems caused by having too few samples, but it introduces variance instead. Unfortunately [Wil00] only implemented the method in a standard ray tracer and not in the context of full global illumination.

[Sun00] describes a spectral framework capable of rendering dispersion. This is done using an extended standard eye tracer. The author extends the ray data structure with information about the monochromacity of the ray, and if the ray is monochromatic, the data structure is also extended with the value of the wavelength. However, since only an eye tracer is used, important phenomena, like indirect lighting, are lost.

Much of the work in for this paper has been inspired by the paper Stratified Wavelength Clusters for Efficient Spectral Monte Carlo Rendering [Eva99]. The authors have implemented an extended bi-directional path tracer ([Laf93]), with a method they call Stratified Wavelength Clustering (SWC). The idea is that emitted light rays carry a cluster of K wavelengths, instead of the more naive approach of just a single wavelength. Only when a ray interacts with a dispersive material are the clusters split into individual wavelengths. The K wavelengths in the cluster are chosen at random, according to the spectral power distribution of the light source. The authors compare their method of emitting clusters of wavelength, against the naive method of emitting a single wavelength at a time. The authors show that SWC converges better than the naive method, while being only marginally more computationally expensive. However, as the authors point out, since they implemented SWC using a bi-directional path tracer, close to pure specular light paths are difficult to model. The authors suggest using photon mapping instead, which as they point out, might cause memory problems because of the relative size of the spectra.

Another work that comes close to what has been done for this paper is [Ieh00]. They implemented a renderer based on photon mapping that was capable of rendering dispersion. They use an adapative representation of the spectral data based on [Rou97]. They also introduce a perceptual error control based on a CIELAB error, which is controlled through a parameter set by the user. In their implementation photons are emitted in two passes; first photons are emitted with an average behavior where even the refraction indices are averaged. Then during rendering, if a caustic is found, non-average photons are re-emitted at certain wavelengths. In this case, some of the paths of the average photons are reused. [Ieh00] shows a lot of promise. However, their method does not easily fit into the standard framework of the photon mapping method and would be difficult to port to a hardware implementation.

All of the methods described above solve Equation 1 either partly or wholly. However, nearly all of them suffer from the problem that the representation of the spectral power distribution is very costly in memory. The main contribution of this paper is a method for solving Equation 1 in an efficient manner, which minimizes memory usage, avoids extra variance, and at the same time is suitable for implementation on modern graphics hardware as described in section 6. However, our method is unable to accurately represent fluorescent spectra.

2 SPECTRALLY BASED PHOTON MAPPING

The method described in this paper has been developed in the context of photon mapping. However, as described in section 6 it appliance goes further than photon mapping.

In general, photon mapping has many steps where the representation of reflection values is important. Since this paper is all about the accuracy and size of the spectral representation, we will go through these steps in photon mapping, and one by one describe how each step has been modified to fit to the subject of this paper. Section 2.1 briefly describes and discusses the representation of the spectral power distribution. Section 2.2 and section 2.3 describes how the photon mapping method [Jen01] have been modified to work within the context of the method described in this paper.

2.1 Representation

Two different strategies have dominated in the subject of representing spectral power distributions; basis functions and point samples.

The first common approach is to represent spectra by the use of basis functions. By pre-analyzing the scene, a set of basis functions and weights that "best" represent the spectra in the scene is found. Each spectrum, is then represented by a set of weights $w_1..w_k$ and a related set of bases $E_1..E_k$. These are summed together in a linear combination to represent the final spectrum, as in Equation 2.

$$\lambda = \sum_{i=0}^{k} w_i(\lambda) E_i(\lambda) \qquad (2)$$

Basis functions have the advantage of being a compact representation, while being able to model complex spectra (i.e. non-smooth). This approach also has the advantage of requiring only a few basis functions to represent smooth spectra well. Often the bases are allowed to be different for each spectrum. Unfortunately this makes spectral multiplication very expensive, as matrix multiplication is an (n^3) operation. To overcome the expensive multiplications, the same basis functions can be chosen to represent all the spectra. This will reduce the cost to O(n). However, unless some spectra are to be misrepresented by too few basis coefficients, a lot of basis functions might be needed to represent all spectra in the scene with sufficient accuracy. The excessive number of basis functions might defeat the entire purpose of trying to overcome the (n^3) cost of the folding operation. [Pee93] presents a method that given a set of spectral power distributions, finds the "best" m basis functions. Here "best" is a measure of the distance between the original spectra and the basis functions. [Pee93] also shows that once the basis functions have been found, the multiplicative cost is comparable to that of point sampling (see description below), while giving a more accurate representation of the original spectra.

Point sampling is a second commonly used method, which is a simple extension of the tristimulus (RGB) method, where a set of n samples is used to represent a spectral power distribution. Point sampling can be seen as a special case of using basis functions, where the bases $E_1..E_k$ are orthonormal to each other.

Point sampling has the advantage, that the folding operation is an O(n) operation. However to model complex spectra, a lot of bins might be needed, which takes up a lot of memory and slows down the multiplication. [Pee93] shows that using Riemann summation for numerical integration of the spectral power distributions, 4 point samples result in less than 5% error of the tristimulus values.

For completeness it should be mentioned that at least two other methods exists.

Sun [Sun00] extends the linear basis approach by introducing a composite model. This model has a dual representation, consisting of a smooth and a spiky part. The smooth part is approximated by a Fourier series, while each of the spikes are represented by a pair (λ_i, w_i), where λ_i gives the location of the i'th spike and wi is a weight. This approach has the advantage of being compact and accurate at the same time. Sun [Sun00] overcomes the inherent O(n^3) time complexity of multiplying two bases and reduces it to O(n) by clever resampling of the Fourier series.

[Rou97] suggests an adaptive representation of the spectral power distribution. All spectral reflectances in the scene are gathered and projected onto a set of basis functions. As basis functions the authors use Haar wavelets, organized by means of binary trees. A user controlled error interval helps control the traversal of the hierachically organized basis functions during rendering. The idea of the method is that for example when rendering dispersion, an error tolerance factor can be used to control the level at which the spectral power distributions are split.

We chose to use uniformly distributed point samples for this paper. There are several reasons for this; multiplication is an O(n) operation and so are the conversions between RGB space and the spectral domain. By using point sampling, reflection values can be kept at full resolution. If we were using basis functions, reflection values would have to be approximated more coarsely than with point samples. In this paper we show that the memory used by point sampling can be cut down to that used by the RGB representation, while still keeping O(n) performance. Thirdly, the core of our algorithm basically consists of matrix multiplications of cost O(N) and is therefore suitable for implementation on modern graphics hardware.

2.2 Photon Mapping, Pass 1: Photon Emission

For this paper, the flux that photons carry can have two different representations; the *full format* and the *compact format*. In the full format, the photon flux is modeled as spectral power distributions, represented as uniformly distributed point samples. We need to understand when and how the compact format is needed, before describing it in full detail in section 2.2.4.

2.2.1 The Full Format

A spectral renderer must solve the rendering equation (Equation 1) on a pr. wavelength basis.

The naive way to solve Equation 1 is to solve it one wavelength at time. This approach increases variance, and therefore also rendering time. As already mentioned, [Eva99] uses light rays carrying clusters of K wavelengths at a time, which decreases variance. However, for this paper we have chosen a third approach, described in the following paragraph.

At the time of emission a photon carries the entire spectrum of the emitting light source. As spectral power distributions are represented as point samples, all N samples are represented in the flux. This increases memory usage linearly with N but also decreases variance in most cases. To see this, assume a photon is hitting a diffuse surface in an epsilon sized area around x and reflecting along ω. As the diffuse BRDF has an uniform probability of reflecting along a given sample, if we were sending out photons carrying only one wavelength at a time, we would on average need to send N times more photons to get the same light intensity along ω than if we sent out a single photon carrying all N samples at once. As long as we are using wavelength-independent BRDFs, except for decreasing variance and thus rendering time, there is no side effect of using N samples instead of $K < N$. By carrying all N samples, the decision of exactly how to reflect/transmit and in which direction is postponed until a wavelength-dependent BSDF is encountered. By using just 1 sample, variance is increased as the decision of which direction to reflect/transmit has already been made before the wavelength-dependent surface is met. For scenes containing wavelength-dependent BSDFs, variance is decreased proportionally with how many photon bounces are needed on average before wavelength-dependent interaction is encountered. The only wavelength-dependent BSDFs used for this paper are those having transmitting effects.

2.2.2 Surface Interaction

When a photon hits a surface, Russian roulette is used to choose either reflection, absorbtion, and if applicable, transmission. If a photon is reflected or transmitted, two important events must be considered. Firstly, a copy of the photon must be made, and tracing of the original continued (see Section 2.2.3). Secondly, the copy of the photon must be stored in the photon map (also Section 2.2.3). For absorbtion, the original photon is discarded.

2.2.3 Tracing and Storing the Photon

When a photon hits a surface, a copy of the spectral photon is made. The copy is stored in the photon map, while the original photon is treated in the following manner:

Upon surface interaction the flux of the photon is modified exactly as described in [Jen01]. If the interaction is a reflection nothing more is done, and we simply keep tracing. However, if the surface interaction is a transmission, then the transmitted direction is wavelength-dependent since we are working within the context of a spectral renderer. This poses a problem, since photons carries N different wavelengths simultaneously and not just one. At the point of transmission, a photon needs to be sent out into each wavelength-dependent direction. This can be done using a recursive ray tracing like technique, Russian roulette, or some third heuristic. The strength and weaknesses of each heuristic is not important in the context of this paper. We have already done a lot of work in this area, and we plan on this being the subject of a later paper. In any case the resulting photon used for tracing along the wavelength represented by the i'th wavelength, will have all samples except the i'th be zero.

It is important to note that all surface interactions are calculated with original spectral data that has not yet been stored in the compact form (section 2.2.4), and thus no loss of data occurs, except that imposed by the discrete sampling representation.

If the photons are stored in RGB format, the flux of the photons will require a meager 4 bytes$*3 = 12$ bytes for each photon. However, if the photons are stored as spectral samples with N entries then the memory requirements will potentially multiply many times. For $N = 100$ the memory requirement will be 400 bytes for storing the flux of a photon. This is 33 times more than the memory requirements for the RGB version.

When photons are stored, the flux they carry is converted to the compact format before being added to the photon map. This compact format is slightly lossy but only uses memory comparable to that of the RGB representation.

2.2.4 The Compact Representation

The compact representation has two different formats. Exactly which format is used depends on whether the flux carried by the photon has more than one non-null sample. As described in section 2.2.3 the flux of a photon which has been through a transmission consists of exactly one non-null sample.

If the flux has exactly one non-null sample there is no need to use N samples for storing the photon, as all samples except one are zero. Instead we use a RGB triplet and store special information in it. The first coordinate is used as a special marker that says this RGB triplet is a not in the standard format, but is a compact spectrum. This is done by setting the entry to -1. This will never go wrong as we need Equation 1 to converge and thus all reflectances must be greater or equal to zero. The second coordinate is used for storing the index of the non-null sample, and the third coordinate for storing the amplitude of the non-null sample. Note that this way of storing the flux does not impose any loss of information.

Note that when each discrete wavelength is represented as exactly one photon, the size of the photon map grows proportionally to N. This fact is not related to our compression method, however. The photon map also grows proportionally to N in the uncompressed version.

If the flux has more than one non-null sample, the spectral flux of the photon is converted to RGB space through the use of Equations 3 to 5.

$$R = k \int_{360}^{830} M_{red}(\lambda) L(\lambda) d\lambda \tag{3}$$

$$G = k \int_{360}^{830} M_{green}(\lambda) L(\lambda) d\lambda \tag{4}$$

$$B = k \int_{360}^{830} M_{blue}(\lambda) L(\lambda) d\lambda \tag{5}$$

The mathematical matching functions $M_{red}(\lambda)$, $M_{green}(\lambda)$ and $M_{blue}(\lambda)$ have been found experimentally using psychological tests. The matching functions used for this paper, have been taken from [hc] [1].

[1] For a more indepth description of the CIE matching functions, see [Fol96].

Using Equations 3 to 5 several spectra might map to the same RGB value. Such spectra are called metamers. During the final gathering step described in section 2.3, photons are collected from the photon map and converted back into the spectral domain. We use [Smi99] for converting from RGB space to the spectral domain. When converting back to the spectral domain, the spectral power distribution might be converted back to a different metamer than the RGB value was originally converted from. More formally, let S be the domain represented by all possible wavelengths, and let C be the domain represented by all RGB colors. Define S_c as the metamer subspace of S that Smits' method [Smi99] maps to. If a spectrum $s_0 \in S \setminus S_c$ is compressed and then converted back to the spectral domain, the resulting spectrum s_1 is clearly not equal to s_0, since s_1 lies in S_c while s_0 does not. This means lossiness is introduced whenever a spectrum not belonging to S_c is compressed. Dispersion is one of the more important cases, where this clearly happens.

Dispersion happens as a light path of type LD^*S and takes a smooth spectrum $s_0 \in S_c$ and maps it to a spiky spectrum s_1 which does not lie in S_c (since the mapping from C to S_c can only produce spectra with more than one non-zero value). Then when s_1 is compacted, it is mapped to color c_1. Then again when c_1 is mapped to S it is not mapped to s_1, but to a spectrum belonging to S_c. This phenomena is shown in figure 1. However,

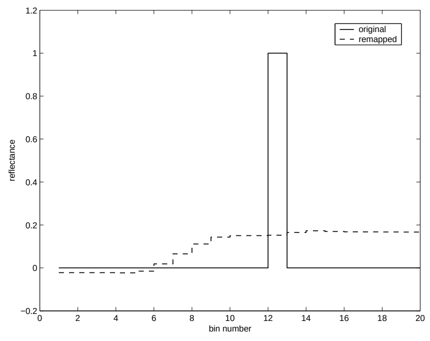


Figure 1: Effect of remapping a spectrum outside S_c

as stated earlier, the particular case shown in Figure 1 has been fixed by storing a photon with exactly one non-null sample in a special compact manner. This is important, as single wavelength spectral power distributions would be destroyed, which would make modelling of wavelength-dependent effects such as caustics difficult. Loss of information can happen in other ways than dispersion. As soon as the product or sum of two spectra lies outside of S_c, the resulting spectrum is lost if it is converted to RGB space and then back to the spectral domain. Algorithm 1 gives the outline on compressing flux when storing a photon in the photon map.

As stated earlier, our method has an inability to represent flourescent spectra, as these are often combinations of a smooth function and sudden spikes, while our method assumes all spectra are either spikes or smooth functions. The straightforward extension of our method to handle flourescent spectra would be to do as in [Sun00]. However, this would cost us the straightforward and efficient hardware implementation that our method is so well suited for (see section 6).

```
Color SelectiveCompacting(SpectralPowerDistribution spd)
if spd.HasOnlyOneNonZeroEntry() then
    Color c;
    c[0] = -1;
    c[1] = spd.IndexOfNonZeroBin();
    c[2] = spd.ValueOfNonZeroBin();
    return c;
end
else
    return spd.AsRGB();
end
```

Algorithm 1: Selective compressing of a spd

2.2.5 Number of Photon Maps

For this paper, we have split the scene into a number of wavelength-independent photon maps as described in [Lar03]. For scenes with lots of wavelength-dependent BRDFs, this idea could perhaps be extended to have photon maps for either each type of BRDF or for each discrete wavelength.

2.3 Photon Mapping, Pass 2: Final Rendering

After the global and caustics maps have been built, the photon mapping method proceeds to the final gathering step. During final gathering, photons are collected from the photon map and used for approximating the Equation 1 in an epsilon sized area around a point *x*.

As the rendering equation is solved spectrally, all compressed photons must be remapped to their spectral representations. How this is done depends on whether the photon represents a single spike or a smooth spectra as described in section 2.2.

Photons in the photon map which represents a single spike , have -1 as the first entry in the RGB triplet describing the flux. To map back into the spectral domain, a spectrum with all samples set to zero is initially created. Then the sample, whose value is represented by the second coordinate in the RGB triplet, has its value set to the reflectance given by the value in the third coordinate of the RGB triplet. This finishes the mapping.

Photons which do not have the first coordinate set to -1, are remapped to the spectral domain using Smits' method [Smi99]. It is during this remapping that a spectral power distribution might be remapped to a different metamer than it originally represented, which means information can be lost.

2.3.1 Locating Photons

It is important to consider exactly how to locate and sum the photons in the photon map. Photons are gathered and summed together exactly as described in [Jen01]. The transformation from the spectral domain to RGB space is a simple integral, and thus linearity assures that it is correct to sum the photons along all wavelengths before conversion to RGB space.

As always, it is important to assure that all photons in the photons map are of similar intensity. This is especially important to be aware of when working with photons that have been through one or more transmissions, as wavelength-dependent transmission might a cause a photon to be reduced to one *N*'th of its original energy.

3 IMPLEMENTATION AND SETUP

To prepare for rendering, we have to make sure that all reflection values in the scene are represented in the spectral domain.

Most artists are used to working with RGB values, so a way of transforming a RGB value to a metamer representation in the spectral domain is needed. We have used the algorithm described in [Smi99] for this purpose. The essence of Smits' method is that it is a fast method of mapping a RGB value to a metamer in the spectral domain. This metamer is represented as *N* uniformly distributed point samples. After all reflection values have been converted to the spectral domain, the photon mapping algorithm is ready to be executed.

Note, that the renderer not only supports loading of materials with RGB values, but also reflection values given as spectral power distribution. This is to make sure that spectral values that lie outside the mapping of Smits' method can be used; one example where this is important could be for using blackbody light sources.

All tests have been performed on a 3.0 GHz Pentium 4 machine, having 1 GB of RAM and running linux. The photon mapping setup used $200,000$ photons, 0.1 as irradiance cache accuracy, 18×57 stratified samples for the final gathering and 200 samples per pixel for the direct lighting. Picture resolution was 128×128. The tests were setup so that the image with the highest number of bins have been used as reference as this should be the most accurate rendering.

4 RESULTS AND DISCUSSION

For testing, we measure the difference in picture quality in dB using the PSNR (*Perceived Signal to Noise Ratio*) measure. Many different definitions of the PSNR exist. The one used for this paper can be seen in Equation 6, in which *MN* is defined as the number of pixels in the image, *a* is the reference image, *b* is the image to be tested, and $a(x,y)$ and $b(x,y)$ are pixel (x,y) in the images.

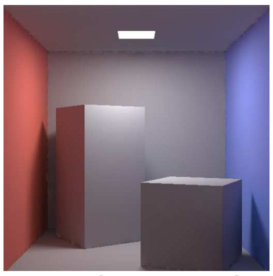
$$PSNR(a,b) = 10log_{10}\left(\frac{MN}{\sum_{x,y}(a(x,y)-b(x,y))^2}\right) \quad (6)$$

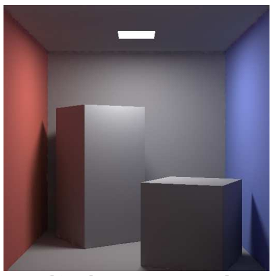
The basis of the test is to try to maximize Equation 6. $PSNR(a,b) = \infty$ means there is no difference between the two pictures. Other interesting measures are the RAM usage and rendering time.

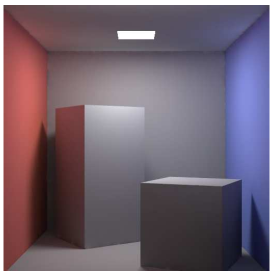
4.1 The Matte Cornell Box

The purpose of the first test is to verify the correctness of the basic idea and implementation. This has been done by creating a matte Cornell box with colored walls and two smaller boxes inside. The verification assumptions are as follows: All reflectances have been defined using RGB colors, the transformation used for mapping between RGB space and the spectral domain is the one described in [Smi99], all illuminants have been defined using RGB colors and finally all BSDFs have behavior independent of wavelength. With this setup, a spectrally based renderer and a RGB based renderer should produce approximately the same image. The tests were run with 4,5,6,8,10,20 and 100 samples for the representation of the spectral power distributions. The tests show very little variance in PSNR numbers, when changing between the compressed and the uncompressed versions or when increasing the number of samples. The only notable result was the savings in ram usage; the reference RGB rendering used 20 MBs, the uncompressed 100 bins version used 103 MBs, while the compressed 100 bins version used 23 MBs. The compressed method has near constant memory usage regardless of bin size. The slight increase in memory usage is due to the irradiance cache, which is still stored in uncompressed format.

Images rendered at a slightly higher resolution than those used for the testings, can be seen in Figure 2. There seems to be nearly no visible difference between any of the pictures in Figure 2. Picture (a) is a little bit brighter, and has a little more pronounced color bleeding than the others, but that is all. Since there is no visible difference between the two pictures we have shown that the spectral renderer works.


(a) 100 bins compressed


(b) 4 bins compressed


(c) 100 bins uncompressed

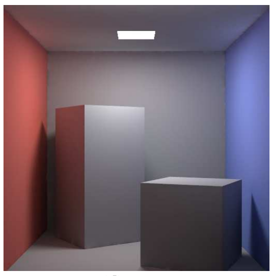

(d) RGB

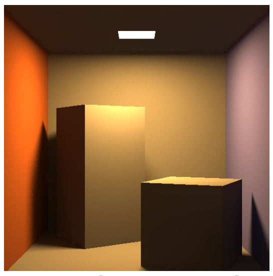
Figure 2: Renderings of the tests made in section 4.1

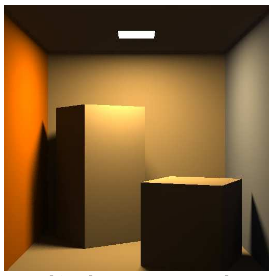
4.2 Spectral Illuminants

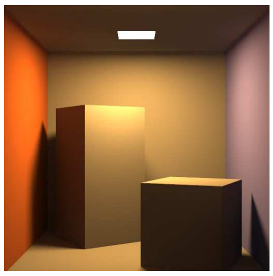
In this test, the light light source from section 4.1 is replaced with a blackbody, which a pure RGB renderer is unable to model correctly. Renderings of the tests are shown in Figure 3, and the test results seen in Table 1. The number of bins clearly matters, as 100 bins give a noticeably different picture than using 4 bins. Due to the underrepresentation of the blackbody spectrum, both the RGB and the 4 bins renderings show strong aliasing in the color of the right wall.
Looking at Figure 3, the renderings of the compressed and uncompressed pictures match closely, which is confirmed by the PSNR values in Table 1.The conclusion is that the compression method renders with approximately the same accuracy as the pure spectral method.

Setup	**PSNR**	**Render Time**	**Ram Usage**
uncompressed			
100 bins	∞	16m 30s	103 MB
20 bins	21.36 dB	6m 39s	40 MB
10 bins	30.46 dB	5m 19s	32 MB
8 bins	31.10 dB	5m 6s	30 MB
6 bins	31.49 dB	4m 54s	29 MB
5 bins	31.58 dB	4m 51s	29 MB
4 bins	28.72 dB	4m 27s	27 MB
RGB	29.63 dB	3m 54s	20 MB
compressed			
100 bins	62.09 dB	17m 6s	23 MB
20 bins	30.48 dB	6m 46s	22 MB
10 bins	30.55 dB	5m 23s	21 MB
8 bins	31.23 dB	5m 11s	21 MB
6 bins	31.54 dB	4m 53s	21 MB
5 bins	31.49 dB	4m 47s	21 MB
4 bins	28.66 dB	4m 28s	21 MB

Table 1: Spectral illuminants test


(a) 100 bins compressed


(b) 4 bins compressed


(c) 100 bins uncompressed

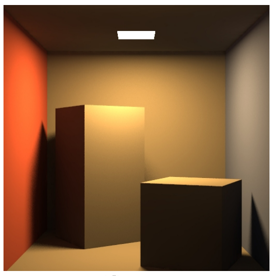

(d) RGB

Figure 3: Renderings of the tests made in section 4.2

4.3 Wavelength-Dependent BSDFs

The last test uses a simple wavelength-dependent glass BSDF, and is used for showing that our method incurs no real loss in rendering quality even when highly specular materials are involved. When calculating transmitted rays, the wavelength of photons in the RGB rendering is assumed to be $550nm$.

Since the test scene involves large caustics, the number of caustic photons has been set to 2 million. The 100 bins test used over 1.3 GB, which is more memory than resided on the test machines.

Thinking a little about the setup, might help to explain the high rendering times given in Table 2. 200 samples have been used for the direct lighting and 18×57 samples for the final gathering. Splitting has been used for the indirect lighting, with a maximum recursion level of 5. For the 100 bins example, whenever a dispersive material is intersected the total number of rays becomes $(18 * 57 + 200) * 2^5 * 100 = 3,923,200$. In the worst case, one ray spawns nearly 4 million extra rays.

Figure 4 shows the test results. Note the extreme variance caused by the close-up rendering of the prism and caustics. The test is difficult as we have an area light source and it shows how many photons are really needed to get good caustics, and how much memory compression matters. This scene really highlights the significance of modelling dispersive behavior. Compared to the other renderings the RGB picture is quite dull. The renderings show significant improvement as the number of bins increases.

Too few bins create aliasing in the form of wrong color reproduction and banding. One example of aliasing in the color reproduction can be seen in the red spot at the front of the prism in the 4 bins rendering. In the pictures created with larger bin sizes this red spot is actually yellow. An example of banding can be seen at the back and front of the prism. The banding becomes more rainbow-like, as the number of bins is increased.

From the PSNR numbers the compressed version with 100 bins is more accurate than the rest of the tests. Reference renderings for the tests can be seen in Figure 4. From Table 2 and Figure 4 it can be concluded that the compressed method gives just as accurate renderings as the pure spectral method. This confirms the theoretical results, as photons which have met dispersive surfaces are stored without any loss of information. The compressed version is faster than the uncompressed one, which is probably due to more memory efficient usage.

Setup	**PSNR**	**Render Time**	**Ram Usage**
uncompressed			
50 bins	∞	2h 34m 14s	794 MB
20 bins	24.21 dB	1h 30m 12s	450 MB
10 bins	24.25 dB	1h 1m 50s	335 MB
8 bins	25.13 dB	58m 42s	321 MB
6 bins	24.54 dB	54m 20s	289 MB
5 bins	23.44 dB	51m 15s	289 MB
4 bins	23.72 dB	51m 31s	266 MB
RGB	22.00 dB	26m 29s	195 MB
compressed			
100 bins	36.46 dB	3h 30m 48s	206 MB
20 bins	24.18 dB	48m 15s	209 MB
10 bins	24.30 dB	29m 45s	195 MB
8 bins	25.13 dB	28m 20s	195 MB
6 bins	24.63 dB	27m 23s	195 MB
5 bins	23.41 dB	26m 56s	195 MB
4 bins	23.67 dB	28m 39s	195 MB

Table 2: Glass BSDF

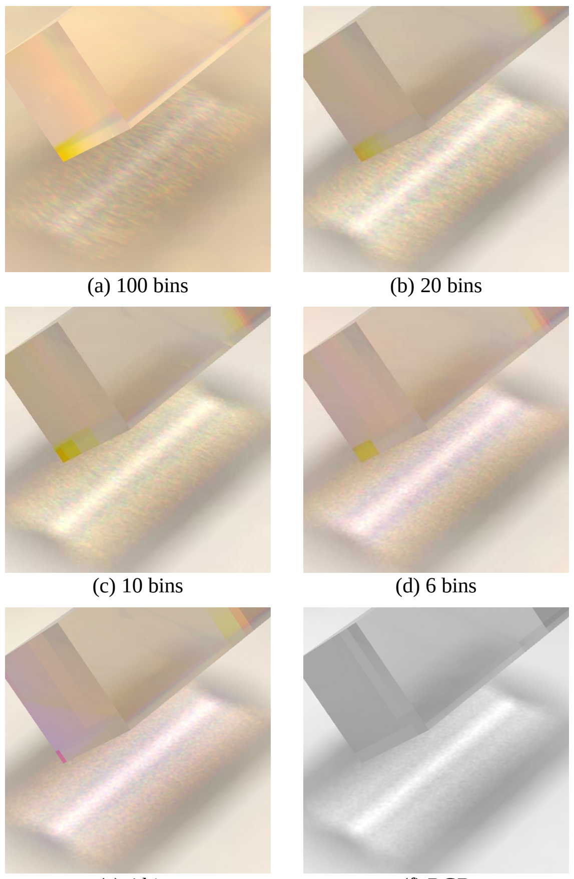

(a) 100 bins (b) 20 bins
(c) 10 bins (d) 6 bins
(e) 4 bins (f) RGB

Figure 4: Renderings of the tests made in section 4.3

5 CONCLUSION

This paper has dealt with the problem of memory inflation when representing light waves as spectral power distributions based on point samples. We have developed a compression method with several advantages:

Firstly, the method is in most cases able to handle spectral materials and light sources, with only a small introduction of error. In other cases, such as with caustics, no errors at all are introduced. An important fact in this context, is that intermediate calculations made before a photon is put into the photon map, are performed without any loss of information.

Because of the memory taken up by the photon map, it has so far been unfeasible to implement spectral photon mapping in which photons carried more than a few wavelengths at a time. Even SWC [Eva99] would be unfeasible with anything but very small clusters. Instead of just solving Equation 1, one or K wavelengths

at a time, the method presented in this paper makes it feasible to solve the equation at full spectral resolution. This reduces variance, and thereby also rendering time. Caustics are calculated at full resolution and without any loss of information. With the method presented in this paper, variance can be reduced, as a photon carry the full spectrum of the light source the first time it hits a transmitting surface, and accounting for the discrete sampling representation, new refracted photons can be directed along all relevant wavelengths at the same time. Contrary to this, if photons represent only a single wavelength, then the direction of transmission around a point x has really been chosen at the time of emission. This increases variance as more photons with different wavelengths will have to be emitted, in the hope of hitting an epsilon sized area around x.
Our method works well for simulating how different lights look in architectural designs and the dispersion effects are useful in movies or visualizations of technical designs.

6 FUTURE WORK

So far the method has only been implemented as a way of saving space when storing photons in the photon map. However, its usefulness goes beyond that. An obvious extension would be to apply it to irradiance caching [War88]. This extends our method to be used with path tracing and bi-directional path tracing, which are methods that benefit from irradiance caching.
If spectral effects are wanted within a realtime renderer, the memory requirements can become a real problem. A 512 by 512 texture takes up 768 KB with 3 bytes pr. pixel (RGB). However in a spectral renderer with 100 samples pr pixel at floating point precision, the memory requirements rise to 400 bytes pr pixel or 100 MBs pr texture, which is clearly not feasible on today's hardware. As the compression method described in section 2.2.4 does not require extra memory and is basically a matrix multiplication, it is straightforward to implement in a hardware shader.
Finally it might be interesting to use [Mey88] when doing the conversion from the spectral domain to RGB space. [Mey88] achieves better color accuracy with fewer wavelengths than the CIE XYZ functions. [Mey88] might allow us to use fewer bins for the spectral power distributions, while still giving acceptable image quality.

ACKNOWLEDGEMENTS

Thanks to Assistant Professor Peter Ørbœk for his support and advice in the inital phase of this work.

REFERENCES

[Arv86] Arvo J.R. Backward Ray Tracing. In ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing, volume 12, pages 259–263. ACM, 1986.

[Col94] Collins S. Rendering crystal glass. In Proceedings of the 2nd Irish Computer Graphics Workshop, TCD-CS-94-20. 1994.

[Eva99] Evans G.F. and McCool M.D. Stratified wavelength clusters for efficient spectral monte carlo rendering. In Graphics Interface, pages 42–49. 1999.

[Fol96] Foley, VanDam, Feiner, and Hughes. Computer Graphics, Principles and Practice, 2nd Edition. Addison-Wesley Publishing Company, 1996. ISBN 0-201-84840-6.

[Gla95] Glassner A.S. Principles of Digital Image Synthesis. Morgan-Kaufmann Publishers, 1995. ISBN 1-55860-276-3.

[Gor84] Goral C.M., Torrance K.E., Greenberg D.P., and Battaile B. Modeling the interaction of light between diffuse surfaces. In Proceedings of the 11t annual conference on Computer Graphics and interactive techniques, pages 213–222. ACM, 1984.

[hc] http://www cvrl.ucsd.edu/. Cvrl color & vision database.

[Ieh00] Iehl J.C. and Péroche B. Adaptive spectral rendering with a perceptual control. In Computer Graphics Forum, volume 19, pages 291–299. 2000.

[Jen01] Jensen H.W. Realistic Image Synthesis Using Photon Mapping. AK Peters, 2001. ISBN 1568811470.

[Kaj86] Kajiya J. The rendering equation. In Proceedings of the 13th Annual conference on Computer Graphcis and interactive techniques, pages 143–150. ACM, 1986.

[Laf93] Lafortune E.P. and Willems Y.D. Bi-directional Path Tracing. In Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93), pages 145–153. 1993.

[Lar03] Larsen B.D. and Christensen N.J. Optimizing photon mapping using multiple photon maps for irradiance estimates. In WSCG 2003 Conference Proceedings. WSCG, Feb 2003.

[Mal86] Maloney L. Evaluation of linear models of surface spectral reflectance with small numbers of parameters. In Journal of the Optical Society of America A: Optics, Image Science, and Vision, volume 3, pages 1673–1683. oct 1986.

[Mey88] Meyer G. Wavelength selection for synthetic image generation. In Computer Vision, Graphics, and Image Processing, volume 41, pages 57–79. jan 1988.

[Pee93] Peercy M.S. Linear color representation for full spectral rendering. In SIGGRAPH Proceedings, volume 27, pages 191–198. 1993.

[Rou97] Rougeron G. and Péroche B. An adaptive representation of spectral data for reflectance computations. In Rendering Techniques '97 (Proceedings of the 8th Eurographics Workshop on Rendering), pages 127–138. Eurographics, 1997.

[Shi98] Shirley and Marschner. Cited by [Smits 1999] as personal correspondence, 1998.

[Smi99] Smits B. An RGB-to-spectrum conversion for reflectances. In Journal of Graphics Tools: JGT, volume 4, pages 11–22. jan 1999.

[Sun00] Sun Y. A spectrum-based framework for realistic image synthesis, ph.d thesis, July 2000.

[Wan04] Wang Q., Xu H., and Sun Y. Practical construction of reflectances for spectral rendering. In Proceedings of the 22th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, volume 12, pages 193–196. WSCG, 2004.

[War88] Ward G.J., Rubinstein F.M., and Clear R.D. A ray tracing solution for diffuse interreflection. In Proceedings of the 15th annual conference on Computer graphics and interactive techniques, volume 22, pages 85–92. August 1988.

[Whi80] Whitted T. An improved illumination model for shaded display. In Communications of the ACM, volume 23, pages 343–349. ACM, 1980.

[Wil00] Wilkie A., Tobler R.F., and Purgathofer W. Raytracing of dispersion effects in transparent materials. In WSCG 2000 Conference Proceedings. 2000.