Compression of Spectral Images Using Spectral JPEG XL

Alban Fichet
Intel Corporation

Christoph Peters
Intel Corporation

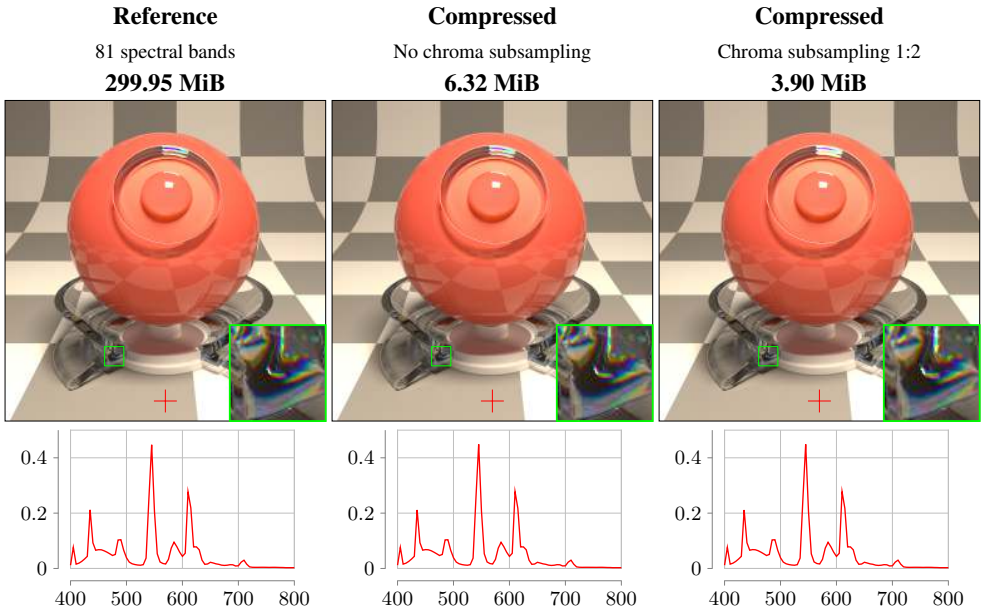


Figure 1. Spectral images have a significant memory footprint because they store a large number of spectral bands. In this paper, we propose a lossy compression scheme that significantly reduces their file size with a negligible loss in quality. We use a cosine transform and separate overall brightness from the shape of the spectrum. The resulting channels are stored using JPEG XL, which provides state-of-the-art lossy compression.

Abstract

The advantages of spectral rendering are increasingly well known, and corresponding rendering algorithms have matured. In this context, spectral images are used as input (e.g., reflectance and emission textures) and output of a renderer. Their large memory footprint is one of the big remaining issues with spectral rendering. Our method applies a cosine transform in the wavelength domain. We then reduce the dynamic range of higher-frequency Fourier co-

 ISSN 2331-7418

efficients by dividing them by the mean brightness, i.e., the Fourier coefficient for frequency zero. Then we store all coefficient images using JPEG XL. The mean brightness is perceptually most important and we store it with high quality. At higher frequencies, we use higher compression ratios and optionally lower resolutions. Our format supports the full feature set of spectral OpenEXR, but compared to this lossless compression, we achieve file sizes that are 10 to 60 times smaller than their ZIP compressed counterparts.

1. Introduction

Spectral rendering is key to accurate color rendition. It describes colors by continuous spectra, which provide a brightness for each wavelength. For any given RGB color, there are many different matching spectra. These are called *metamers*, and in a renderer it makes a difference which metamer is used. For example, when different metameric light sources illuminate the same surface, the reflected color may be perceived differently because the metameric spectra get multiplied by the spectrum for the albedo of the surface. RGB renderers typically pretend that all light consists of only three different wavelengths and cannot account for metamers correctly. Thus, spectral rendering offers a tangible advantage. The wavelength domain becomes yet another dimension for integration, which can be handled with fairly low overhead using techniques such as hero wavelength spectral sampling [Wilkie et al. 2014]. Therefore, it is gaining traction in production rendering.

However, the transition to spectral rendering has a big impact on storage requirements because RGB triples are replaced by more expressive spectral representations. Renderers use several different kinds of spectral data, and all of these may need to be stored in high-resolution images with a spectrum per pixel. Emission spectra describe colors of light sources. They typically have high dynamic range and may have a large number of sharp peaks, which calls for dense sampling. Reflectance spectra describe how the albedo of a surface depends on the wavelength. These are bounded between zero and one to satisfy energy conservation, and measured data show that they are rather smooth. Fluorescent surfaces absorb light at one wavelength and immediately reemit it at another wavelength. That calls for bispectral data, i.e., for functions that depend on two wavelength parameters. Finally, the rendered image can be stored in a spectral fashion, which makes it agnostic to particular color spaces or camera response curves. These images essentially behave like emission spectra.

Fichet et al. [2021] store these data using samples at many wavelengths. They established conventions for how to name channels in OpenEXR files to ease the exchange of all sorts of spectral data between applications. However, OpenEXR only supports basic lossless or lossy compression of its floating-point channels, and the number of channels in these spectral images tends to be large. Therefore, spectral EXR files are generally much larger than their RGB counterparts.

As remedy for this situation, we propose a simple scheme for lossy compression of spectral images. To exploit redundancy across different spectral channels, we take a cosine transform in the wavelength domain.All higher-order Fourier coefficients get divided by the DC component (i.e., the mean) to squash their dynamic range. These channels are then stored using JPEG XL, which provides state-of-the-art lossy image compression. Optionally, the resolution may be reduced for higher-order Fourier coefficients, which essentially performs chroma subsampling. We provide a compression and decompression utility that converts between spectral OpenEXR and our spectral JPEG XL, retaining all channels and metadata. Our evaluation on a variety of spectral images shows that we typically reach compression ratios compared to spectral OpenEXR with ZIP compression ranging from $10\times$ to $60\times$ without introducing noticeable error.

2. Related Work

Until recently, there were no clearly standardized formats for spectral images in rendering. A common practice is to store one RGB image per spectral channel [Yasuma et al. 2010]. Domain-specific formats such as ENVI for satellite data are clearly standardized but lack features that are relevant for rendering such as support for bispectral data or polarization [ENVI]. Therefore, it has been difficult to exchange spectral data between different renderers.

Spectral OpenEXR [Fichet et al. 2021] establishes a standard to alleviate this problem. The basic approach is to store continuous spectra by providing samples for sufficiently many wavelengths. All these samples are stored in a single OpenEXR file using one channel per wavelength sample. The channel names are standardized to organize them across different layers, providing the wavelength and some meta data. In this manner, the format supports spectral data, bispectral data, emission, reflectance, polarization, and stereoscopic images. It also includes RGB channels to offer a meaningful preview in standard EXR viewers.

The format inherits its compression methods from OpenEXR. There are three approaches for lossless compression [OpenEXR]: PIZ applies a wavelet transform and Huffman coding, ZIP applies zlib to differences of adjacent pixels, and RLE uses run-length encoding on such differences. Additionally, there are two simple lossy compression schemes [OpenEXR]: PXR24 discards the least significant 8 mantissa bits of 32-bit floats before applying ZIP, and B44 achieves a fixed compression rate by packing blocks of 4×4 half-precision floats into 14 bytes. All of these methods roughly halve the file size in common use cases [OpenEXR].

For low dynamic range RGB images, lossy compression methods achieve much higher compression ratios. Early image compression formats (e.g., GIF) use color palette reduction, both because of the limitation of the display devices of the time and

for storage efficiency. Later, JPEG [ITU 1992] offered lossy compression using the discrete cosine transform of 8×8 tiles and separation of luminance and chrominance. A quantization matrix diminishes the accuracy of the resulting coefficients, usually saving more on higher frequencies, before run-length encoding and entropy coding are applied. The used color space enables additional lossy compression by means of downsampling of chrominance channels, a practice known as *chroma subsampling*. This practice works well since human observers are less sensitive to spatial variations in chrominance. JPEG XL is an evolution of JPEG [ISO 2024]. It improves compression ratios, adds the option for lossless compression, and supports high dynamic range images. We use it as the foundation of our spectral format with lossy compression.

Spectral upsampling makes these image formats applicable for spectral rendering. The goal is to convert RGB triples into plausible reflectance spectra, which match the RGB triple. An option for sRGB is to combine hard-coded spectra in a piecewise-linear fashion [Smits 1999]. Alternatively, a lookup table can turn RGB triples into coefficients of a suitable three-dimensional family of smooth functions [Jakob and Hanika 2019]. It is also possible to characterize spectra by their Fourier coefficients. A nonlinear reconstruction method known as *bounded MESE* (maximum entropy spectral estimate) turns these Fourier coefficients into valid reflectance spectra with values in $[0, 1]$ [Peters et al. 2019]. RGB triples can be converted to three Fourier coefficients, but unlike other representations, this method also scales to more accurate spectral representations with more than three coefficients.

Satellite imagery also uses spectral images extensively. Due to the large size of the raw data, lossy spectral compression is a powerful asset, reducing traffic from the satellite to a ground station on a constrained downlink bandwidth. Previous work on lossy spectral image compression mostly targeted this application. Kaarna [2007] gave a summary of different lossy compression techniques. Our spectral decomposition is similar in spirit to the discrete cosine transform (DCT)-based approach of Abousleman et al. [1995]. The authors proposed a 3D DCT encoding on $8 \times 8 \times 8$ blocks or a 2D DCT combined with predictive coding (DPCM, or differential pulse-code modulation) on the spatial and spectral domain.

Our work rather focuses on spectral images for computer graphics. This allows us to reuse lossy compression techniques designed for RGB image compression, such as lowering chrominance resolution found in AC components and allocating a varying compression ratio depending on the signal frequency. Our work does not propose a new lossy compression scheme for single-channel images. Instead, we fully rely on JPEG XL for the lossy compression. Taking inspiration from Peters et al. [2019], we utilize a cosine transform of spectra to feed JPEG XL with framebuffers for lossy compression and control the compression quality of each framebuffer.

3. Spectral JPEG XL

We now introduce our spectral JPEG XL format. The goal is to cover the full feature set of spectral OpenEXR while offering much higher compression ratios at the cost of some loss in quality. To benefit from the fact that spectra are often low-frequency functions, we compute Fourier coefficients of spectra. When there are large brightness variations in the image, the resulting coefficients will all have high dynamic range. We diminish this dynamic range for all but one coefficient by dividing all higher-order coefficients by the coefficient for frequency zero (Section 3.1). Then we store each individual coefficient using JPEG XL (Section 3.2). We have an automated method to determine how to trade quality against file size for different coefficients in an automated fashion (Section 3.3). Our format supports chroma subsampling (Section 3.4), supports bispectral images (Section 3.6), and preserves the metadata of spectral OpenEXR (Section 3.7). Our reference implementation converts back and forth between spectral OpenEXR and spectral JPEG XL (Section 3.8).

3.1. Computing Quantized Fourier Coefficients

Consider a spectral image with samples at the sorted wavelengths $\lambda_0, \dots, \lambda_{n-1} \in \mathbb{R}$. For a single pixel, we have values $g_0, \dots, g_{n-1} \in \mathbb{R}$. For reflectance spectra as well as many emission spectra, this signal will be rather low frequency. We compute Fourier coefficients to be able to exploit this redundancy in the samples. A common discrete cosine transform treats each sample like a Dirac-δ pulse at its wavelength. This notion does not reflect the smooth nature of most real spectra and is incompatible with existing reconstructions of spectra from Fourier coefficients [Peters et al. 2019]. Instead, we compute a cosine transform for the signal that arises from linear interpolation.

To keep our exposition self-contained, we reproduce the corresponding formulas here [Peters et al. 2019]. First, we have to map wavelengths to phases for all $l \in \{0, \dots, n-1\}$:

$$\varphi_l := \pi \frac{\lambda_l - \lambda_0}{\lambda_{n-1} - \lambda_0} - \pi \in [-\pi, 0].$$

Note that we only use one half of the domain $[-\pi, \pi]$ since the signal is implicitly mirrored on the x-axis. That means that we are using a cosine transform instead of a periodic Fourier transform. That is equivalent to computing Fourier coefficients for a signal on $[-\pi, 0]$ that has been mirrored with regards to the y-axis to make it even. The interpolated signal on the interval $[\varphi_l, \varphi_{l+1}]$ has the following gradient and y-intercept:

$$a_l := \frac{g_{l+1} - g_l}{\varphi_{l+1} - \varphi_l}, \qquad b_l := g_l - a_l \varphi_l. \tag{1}$$

We get Fourier coefficients by solving an integral in closed form. The result for frequency zero is

$$c_0 := \frac{1}{\pi} \sum_{l=0}^{n-2} \left[\frac{a_l}{2} \varphi^2 + b_l \varphi \right]_{\varphi_l}^{\varphi_{l+1}} \in \mathbb{R}. \tag{2}$$

The Fourier coefficient for frequency $j \in \{1, \ldots, n-1\}$ is

$$c_j := \frac{1}{\pi} \Re \sum_{l=0}^{n-2} \left[\left(a_l \frac{1 + ij\varphi}{j^2} + b_l \frac{i}{j} \right) \exp(-ij\varphi) \right]_{\varphi_l}^{\varphi_{l+1}} \in \mathbb{R}, \tag{3}$$

where $\Re$ denotes taking the real part of a complex number.

Note that we compute a total of n real Fourier coefficients. Our reasoning is that we want to be able to get an exact reconstruction of all spectral bands out of the Fourier coefficients as long as they are stored without error. We could use the (bounded) MESE [Peters et al. 2019], but that would reconstruct a continuous spectrum with slightly mismatching values at $\lambda_0, \ldots, \lambda_{n-1}$. This source of error is easily avoided. Equations (1), (2), and (3) map $g := (g_0, \ldots, g_{n-1}) \in \mathbb{R}^n$ to $c := (c_0, \ldots, c_{n-1}) \in \mathbb{R}^n$ in a linear fashion. We can feed this transform with the canonical basis vectors $e_0, \ldots, e_{n-1} \in \mathbb{R}^n$ to get the column vectors of a matrix $A \in \mathbb{R}^{n \times n}$ such that $c = Ag$. Then it is easy to reconstruct the sample values from exact Fourier coefficients: $g = A^{-1}c$. We use this formula for decompression. Note that the inverse matrix A^{-1} only needs to be computed once per spectral image. The cost is negligible.

In images with large brightness variation, all Fourier coefficients will exhibit a similar spatial pattern with high dynamic range. Naturally, that makes them difficult to compress on their own. To alleviate this problem, we divide $c_1, \ldots, c_{n-1}$ by c_0. Since spectra are expected to be positive, $c_0 = 0$ also implies $c_1, \ldots, c_{n-1} = 0$; therefore, it is easy to avoid division by zero with a branch:

$$d_j := \begin{cases} \dfrac{c_j}{c_0} & \text{if } c_0 \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

In any case, $c_j = d_j c_0$. The Fourier coefficient c_0 is the average value across the whole spectrum, which is a natural way to define overall brightness (cf. Figure 3). It does not correspond directly to human perception of brightness, but at least it behaves in a similar fashion. Thus, the division essentially normalizes spectra for equal brightness, which should reduce the dynamic range in the coefficients considerably.

From this point onward, we treat $d_1, \ldots, d_{n-1}$ as low dynamic range data. For each frequency $j \in \{1, \ldots, n-1\}$, we compute the minimal and maximal values of d_j across the whole image, $d_{j,\min}, d_{j,\max}$. Then we turn the Fourier coefficients into values between zero and one:

$$f_j = \frac{d_j - d_{j,\min}}{d_{j,\max} - d_{j,\min}} \in [0, 1].$$

For decompression, we undo this transform using $d_j = f_j(d_{j,\max} - d_{j,\min}) + d_{j,\min}$. If it is beneficial to the compression, we can quantize these normalized values to a fixed bitrate.

3.2. Compressing Image Data Using JPEG XL

Thus far, we have only transformed spectra. The next step is to store them in a way that achieves lossy compression with good compression ratios and quality. Once per image, we have to store $d_{j,\min}, d_{j,\max}$ for $j \in \{1, \dots, n-1\}$. Additionally, for each pixel we store $c_0 \in \mathbb{R}$ (which might have high dynamic range) as well as $f_1, \dots, f_{n-1} \in [0, 1]$. The file also has to hold some metadata such as $\lambda_0, \dots, \lambda_{n-1}$; see Section 3.7.

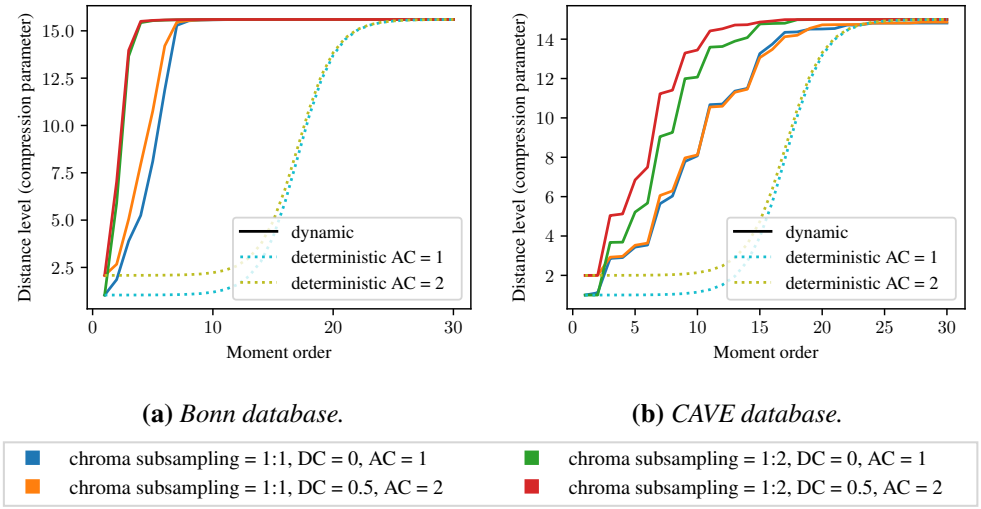
We store these data using the royalty-free JPEG XL image format [ISO 2024]. Its state-of-the-art lossy and lossless image compression techniques can handle many framebuffers with different scalar types (e.g., float, half, or fixed-point with varying bit counts) within the same file. It also enables us to store metadata in so-called boxes. However, the reference implementation libjxl has not reached a stable version yet. Some features are missing and others have issues. We discuss our workarounds throughout this paper but hope that they will eventually become superfluous through a better libjxl.

In principle, JPEG XL supports having one main image and up to 255 sub-images, which sounds like a good match for c_0 and $f_1, \dots, f_{n-1}$. Unfortunately, the current implementation in libjxl does not allow us to tweak the compression ratio and subsampling on a per-sub-image basis. Due to these limitations, we currently use one JPEG XL file per channel so that we have full control over the compression parameters. The main file stores c_0 and all metadata. This file provides a natural way to preview spectral images in any software with JPEG XL support, since it shows the overall brightness. Our implementation uses 32-bit floats for c_0 when the spectral OpenEXR file uses them and 16-bit floats otherwise. The remaining $n-1$ files store $f_1, \dots, f_{n-1} \in [0, 1]$ as low dynamic range images. We always set the parameter for the compression effort of JPEG XL to 7, which is the default value in libjxl.

3.3. Optimal Compression Curves

The main parameter to trade between quality and file size in JPEG XL is the distance level. It controls the psychovisual error metric Butteraugli, which is used by JPEG XL to assess compression error. Exposing this parameter to the user for each frequency $j \in \{1, \dots, n-1\}$ would be nonintuitive. At the same time, it is not reasonable to use the same value across all frequencies. For smooth spectra, high-frequency coefficients have lower variation prior to renormalization, and in rendering they usually have less impact on final colors. To provide a more intuitive control, we propose to expose two quality parameters, one for the high dynamic range DC component (i.e., frequency zero) and another for the AC components (higher frequencies). We determine all other distance levels in an automated fashion.

The distance level for f_1 is controlled by the user directly. We compress f_1 using this distance level. Then we decompress and compute the root-mean-square error



(a) *Bonn database.* **(b)** *CAVE database.*

Figure 2. The average of the compression curves across each of our two databases of spectral images. Dynamic curves are produced by our optimization procedure; deterministic curves use Equation (5). Curves for individual images can be found in the supplemental materials. The legend states the used subsampling ratios and distance levels.

(RMSE) that we get when all Fourier coefficients except for f_1 are stored exactly (see Equation (6)). We can also evaluate this RMSE for a compressed version of f_j with $j \in \{2, \dots, n-1\}$ (using exact versions of all other coefficients). Our goal is to achieve a similar contribution to the error from each frequency, while using greater distance levels for higher frequencies. We compress and decompress each coefficient image f_j repeatedly inside a binary search to find distance levels that come close to this goal. The resulting distance level for each frequency is our compression curve.

The repeated compression and decompression in this procedure is costly. For images of size $512 \times 512 \times 31$, we observe computation times between 20 and 75 seconds. The resulting compression curves are fairly diverse (see Figure 2). A simple alternative is to use a flat compression curve across all AC components, but that is suboptimal. As a middle ground between these two extremes, we propose a hard-coded compression curve. It is a shifted sigmoid function $S(x)$ starting at the user-provided value L_1 and ending at the maximal allowed distance level $L_{n-1} = 15$:

$$S(x) := \frac{1}{1 + \exp(-x)}, \tag{4}$$

$$L_j := L_1 + \frac{S\left(10^{\frac{2j-n}{n}} - 1\right) - S\left(10^{\frac{2-n}{n}} - 1\right)}{S\left(10^{\frac{2(n-1)-n}{n}} - 1\right) - S\left(10^{\frac{2-n}{n}} - 1\right)} (L_{n-1} - L_1). \tag{5}$$

Our optimization of compression curves is inspired by the quantization of Fourier coefficients in related work [Rapp et al. 2022]. We also tried using the method pro-

(a) *Spectral image converted to RGB.*

(b) *DC component* c_0.

(c) *Normalized AC components* d_j *converted to RGB.*

Figure 3. We normalize the AC components of the signal by dividing out the DC component. The DC component (b) is the mean across wavelengths and roughly corresponds to overall brightness. The normalized AC components (c) correspond to the chrominance.

posed in this work for quantization of our Fourier coefficients. However, lossy JPEG XL compression does not seem to benefit from more aggressive quantization. For this reason, we use a fixed high number of bits for quantization.

3.4. Chroma Subsampling

The human visual system is far more sensitive to spatial variation in luminance than in chrominance. Thanks to the division by the DC component c_0, our spectral representation has a separation of luminance (or more precisely, mean energy across the spectrum) and chrominance built in (Figure 3). In addition to the methods presented thus far, we can bring down the storage cost for the AC components $f_1, \ldots, f_{n-1}$ by simply reducing their resolution. For our experiments, we halve the horizontal and vertical resolution using the up- and downscaling provided by libjxl. With our parameters, this implementation uses a 12×12 kernel for downsampling. It also reduces ringing through clamping based on original image values, and it is designed to work well with the upsampling method in libjxl. This method gives an appreciable file size reduction. Error metrics like the RMSE go up significantly, but perceptually the errors are more acceptable.

3.5. RGB Channels

Spectral OpenEXR supports storing RGB channels so that any software with EXR support will display a meaningful preview of the spectral data. These channels have a high dynamic range, and storing them explicitly in spectral JPEG XL would incur a large cost. Instead, we argue that the image holding c_0 has to be enough as a preview. Upon compression, we discard RGB channels but store whether they were present as metadata. If they were present, they are recreated from the decompressed spectra

using the definition of the color space indicated in the metadata. Then the spectral OpenEXR file coming out of the decompression procedure can be previewed just like the original file.

3.6. Bispectral Images

Bispectral images are still seldomly found, but since spectral OpenEXR supports them, we also want to cover this use case in a simple fashion. Like spectral OpenEXR, we treat a bispectral image as a stack of spectral images. The first one of those holds the diagonal of the reradiation matrix, where both wavelengths are equal. It corresponds to reflectance without fluorescence, and therefore it is stored in a reflective layer ("T"). Then there is one layer per incoming wavelength λ_i providing reradiation spectra for a monochromatic illuminant ("T.$\lambda_i$nm").

3.7. Metadata

We use the JPEG XL box mechanism to hold all metadata of the original spectral OpenEXR file in our Spectral Graphics Extended Group (SGEG) box. It holds information about the scene, capture, photographic metadata, or render presets. Additional spectral OpenEXR metadata can characterize the spectral data in terms of radiometric units or spectral response curves. Some metadata about our spectral JPEG XL format come on top of that. Overall, the SGEG box stores the following information:

- The revision tag of the file format to allow future improvements,
- The number of JPEG XL files needed to store all framebuffers,
- Information for each spectral image,
- Information for each additional OpenEXR framebuffer,
- A bitstream containing the original OpenEXR attributes.

Since the OpenEXR layout for spectral data allows storing multiple spectral images, e.g., stereo images, we can have multiple spectral images represented by a single spectral JPEG XL file collection. Spectral OpenEXR images may also contain additional non-spectral framebuffers, although we remove the RGB buffers to save space. Each spectral image has the following metadata:

- The root name, e.g., "left.S0" for the left image of an emissive image,
- The indices of the relevant framebuffers,
- The sampled wavelengths $\lambda_0, \ldots, \lambda_{n-1}$,
- The minimal and maximal values for each frequency $d_{j,\min}, d_{j,\max}$.

For standard OpenEXR framebuffers, we store:

- The corresponding layer name in the original OpenEXR file,
- The framebuffer index in the JPEG XL file.

3.8. Reference Implementation

We provide an open source C++ implementation of our compression method as supplemental using OpenEXR 3.3.2 and libjxl 0.11.1. It provides two executables:

- `compress` converts a spectral OpenEXR image into a compressed spectral JPEG XL, potentially with multiple views and additional framebuffers treated as gray images.
- `decompress` allows for the decompression of a spectral JPEG XL image into an OpenEXR image, including additional framebuffers and original metadata.

The compression utility takes command-line parameters for the distance level for c_0 and the quantization, distance level, and subsampling for f_1. Additional arguments determine whether the compression curve is flat, deterministic (Equation (5)), or dynamically optimized. Since lossy JPEG XL does not benefit from quantization, we recommend keeping the quantization flat at a high bitrate. The two distance levels are the main parameters to control the file size (bigger values mean smaller files). We recommend keeping the DC component c_0 lossless (distance level 0) or nearly lossless (distance level 0.5). Reasonable values of the distance level for f_1 are between 0 and 3, and the deterministic compression curve gives fast compression with a reasonable tradeoff between quality and file size. Compression times can be long, especially with optimization of compression curves, but decompression is fast and the computation time barely depends on the chosen parameters.

4. Results

To evaluate our methods, we use two databases of measured spectral images (Section 4.1). We systematically analyze the impact of different parameter choices on the quality of our results (Section 4.2). Then we compare our method to a simple approach where JPEG XL is applied to each spectral band directly (Section 4.3). We also provide statistics on the quality and compression ratio for all images in our database (Section 4.4) and detailed results in the supplemental material. Finally, we discuss the run time of our compression and decompression method (Section 4.5).

4.1. Databases of Spectral Images

We evaluate our compression method using two databases of spectral images. The CAVE database [Yasuma et al. 2010] consists of 32 still photographs of various ob-

jects with controlled indoor lighting divided into five sections: "stuff," "skin and hair," "paints," "food and drinks," and "real and fake." The authors provide them as PNGs with 16-bit unsigned integers per band. `watercolors_ms` is an exception with 8 bits, and therefore we discard it. The resolution is 512×512.

The Bonn database [Merzbach et al. 2019] contains 13 material samples measured using an X-Rite Tac7. We use the spectral albedo and specular maps out of these measurements, providing a set of 26 spectral images. These have been generated from RGB maps that come out of the svBRDF fitting procedure, but the spectral upsampling method is overfitted to the measured spectra for each individual material [Peters et al. 2019]. The resolution varies from 512×512 to 2048×2048.

With these two databases, we have emission spectra in the CAVE database as well as reflectance spectra in the Bonn database. For both of them, the wavelengths range from 400 to 700 nm with 10 nm increment (31 bands). We have converted all of these images to spectral OpenEXR with 32-bit floats. Additionally, we use a rendered spectral image with a fluorescent illuminant and 81 bands (Figure 1).

4.2. Impact of Compression Parameters

Our method exposes three parameters to control the quality: the distance level for the DC component, the distance level for the first AC component, and the chroma subsampling ratio. We will now investigate the impact of these parameters on compression ratios and quality one by one to arrive at recommended configurations used in the remainder of our evaluation.

The distance level for the DC component c_0 has a strong perceptual impact. The DC component captures the mean brightness of the spectrum and thus is closely related to luminance. Figure 4 shows that even moderately high values lead to a clear degradation in quality. We recommend keeping the DC component lossless (distance level 0) or using a small distance level of 0.5.

The distance level for the first AC component has a weaker impact on the quality. Figure 5 shows that artifacts only become visible in RGB images at extreme values. Figure 6 shows corresponding spectra for select pixels. We observe that spectra for flat image regions are preserved well (red and blue graphs). The green pixel is in a region with high-frequency spatial variation, and therefore the higher-frequency Fourier coefficients also have greater error, which manifests as high-frequency noise (green graphs). These results use our deterministic compression curve (Equation (5)). Since these curves always reach the maximal distance level of 15 eventually, we recommend using moderate values as the starting point. In our experiments we use a distance level of 1 or 2.

Chroma subsampling brings the file size of AC components further down by reducing their resolution. Figure 7 shows the impact for different subsampling ratios. The quality degradation for a ratio of 1:2 is perceivable in magnified insets but

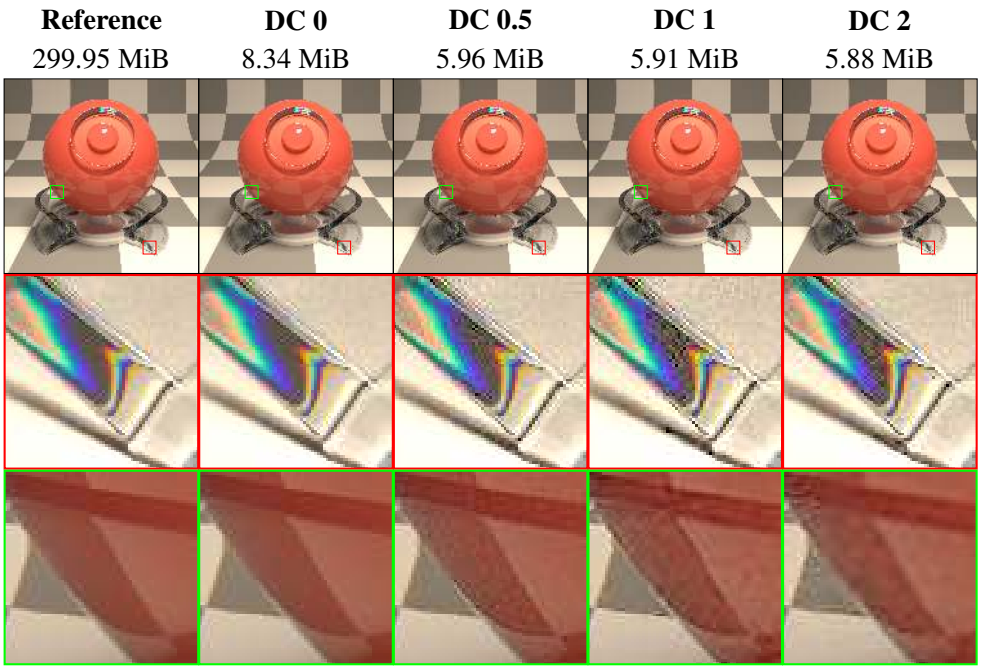


Figure 4. Increasing the distance level for the DC component leads to perceivable artifacts quickly. We use deterministic compression curves starting at $L_1 = 1$ and no chroma subsampling.

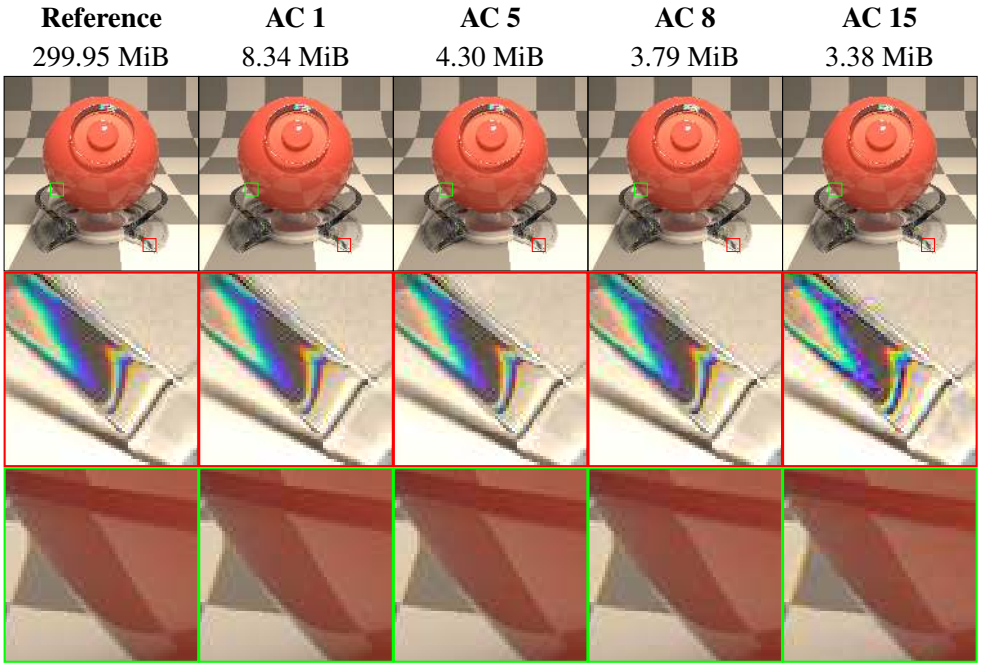


Figure 5. High distance levels are acceptable for the first AC component. The compression curve is deterministic (Equation (5)), DC is lossless, and we do not use chroma subsampling.

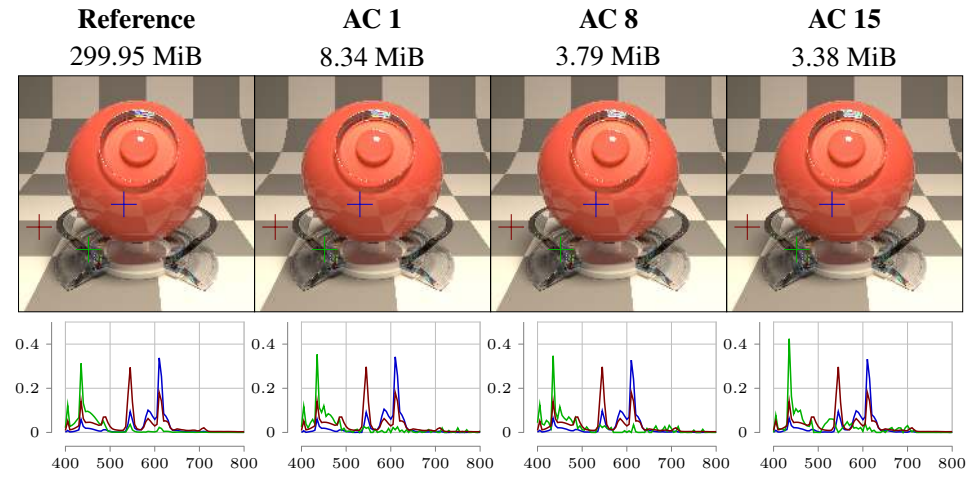


Figure 6. Spectra for three highlighted pixels from the decompressed spectral images in Figure 5. In flat image regions, spectra are preserved well, but high-frequency image regions incur high-frequency noise in the wavelength domain.

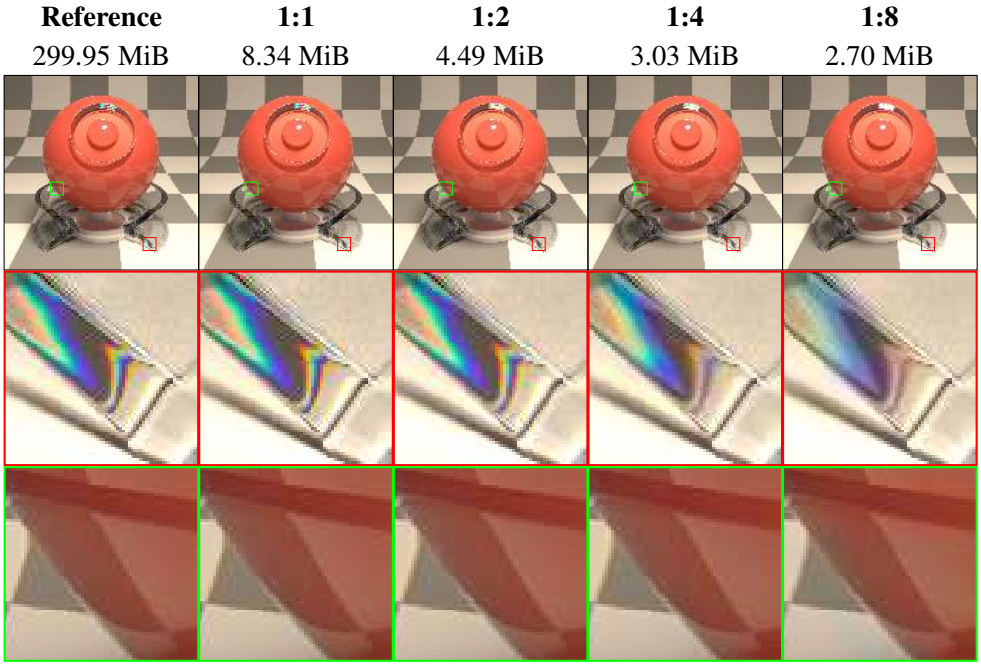


Figure 7. Comparison of different subsampling ratios. The DC component is lossless and we use a deterministic compression curve with $L_1 = 1$.

should be unproblematic in many cases. At higher ratios, the fidelity of color transitions diminishes noticeably. For most cases, we recommend sticking to 1:1 or 1:2. Whether 1:2 subsampling is acceptable depends on the use case. For example, the Bonn database has many sharp color transitions, and our error metrics increase substantially when enabling chroma subsampling (see Section 4.4).

In summary, the distance level for DC should be 0 or 0.5, reasonable distance levels for the first AC component are 1 or 2 combined with deterministic or dynamic compression curves, and 1:2 chroma subsampling may be used when that quality degradation is acceptable.

4.3. Comparison to Simple JPEG XL Compression

A simpler approach compared to our method is to store each spectral band as a JPEG XL file directly. The number of files will be the same, and while there is no preview of overall brightness, the spectral bands themselves also provide a reasonable preview. Figure 8 compares this simple method to our approach. Unsurprisingly, the simple method introduces greater error to the overall brightness and contrast of images, which manifests as excessive blur and blocky ringing artifacts. With our method, we have explicit control over the compression ratio of the mean brightness and can preserve this important quantity more faithfully.

4.4. Quality Statistics

Our supplemental document provides an extensive collection of results and statistics for all images in our database. Here we provide a summary of these results. For this purpose, we need a scalar-valued image error metric. In general, perceptual metrics are preferable, but for spectral images there are no agreed-upon standards for such metrics and what is meaningful depends on the use case. Therefore, we use a simple RMSE:

$$\mathrm{RMSE}(g, g') := \sqrt{\frac{\sum_{x=0}^{w-1}\sum_{y=0}^{h-1}\sum_{l=0}^{n-1}\left(g'_{x,y,l} - g_{x,y,l}\right)^2}{whn}}, \tag{6}$$

where $g_{x,y,l}$ and $g'_{x,y,l}$ provide images of width w, of height h, and with n spectral bands.

Figure 9 provides averaged statistics and scatter plots for all images in the two databases and for different techniques and configurations. The reported compression ratios are relative to spectral OpenEXR with lossless ZIP compression and without RGB channels. We observe that the simple method performs relatively well in terms of RMSEs, which is unsurprising since the RMSE rewards low errors on each channel individually. Still, our method achieves similar RMSEs at similar compression ratios but also scales to higher compression ratios. Chroma subsampling increases compression ratios considerably, especially in combination with lossy compression of the DC

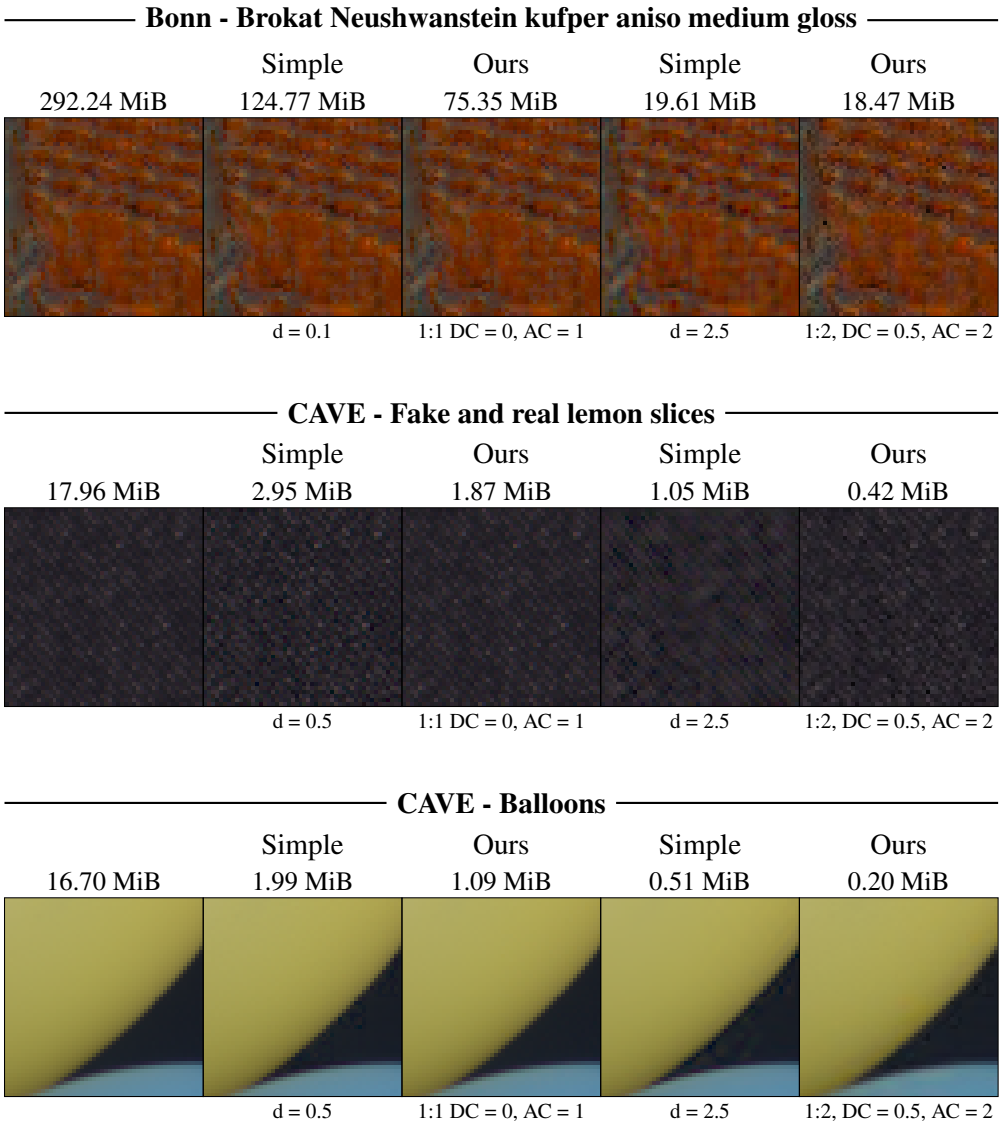
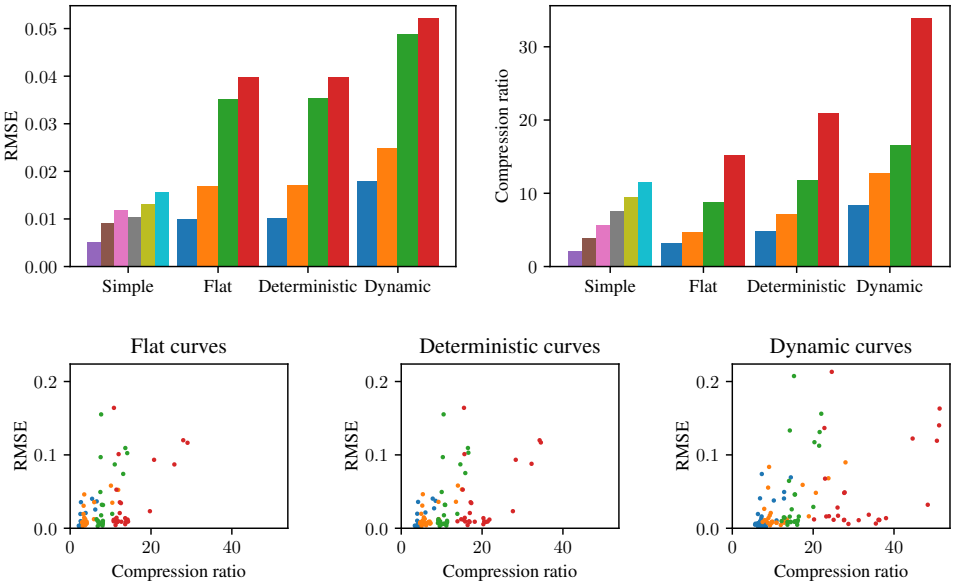
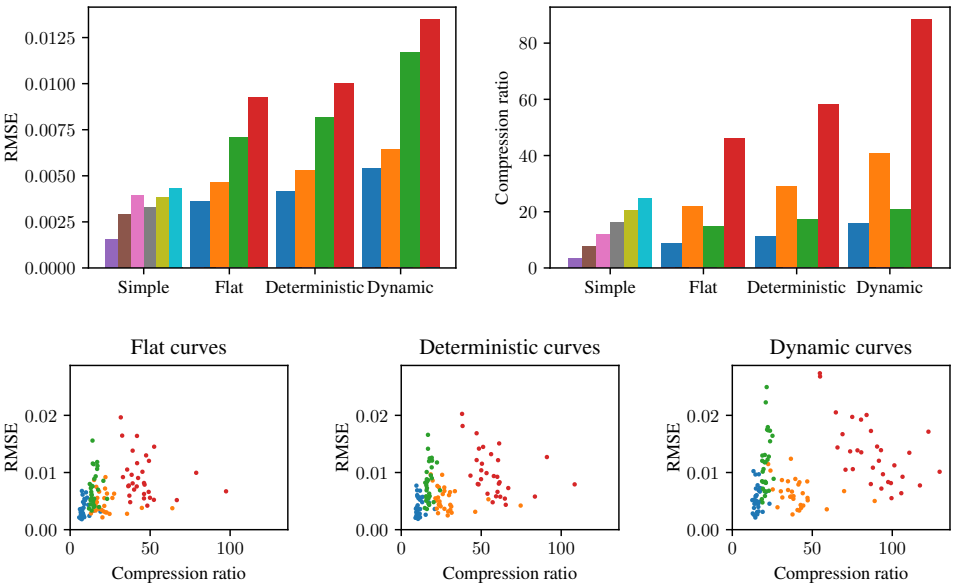


Figure 8. Comparison of our compression method to a simple method, which stores each spectral band as a JPEG XL file. We show 50×50 crops to illustrate common artifacts. The used distance levels and chroma subsampling ratios are stated below each image. Our method uses deterministic compression curves here (Equation (5)). File sizes refer to the full image. The simple scheme shows typical JPEG compression artifacts in the luminance. Our method compresses the mean brightness less aggressively and thus preserves such features more faithfully.



(a) *Bonn database.*



(b) *CAVE database.*

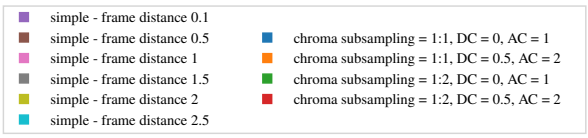
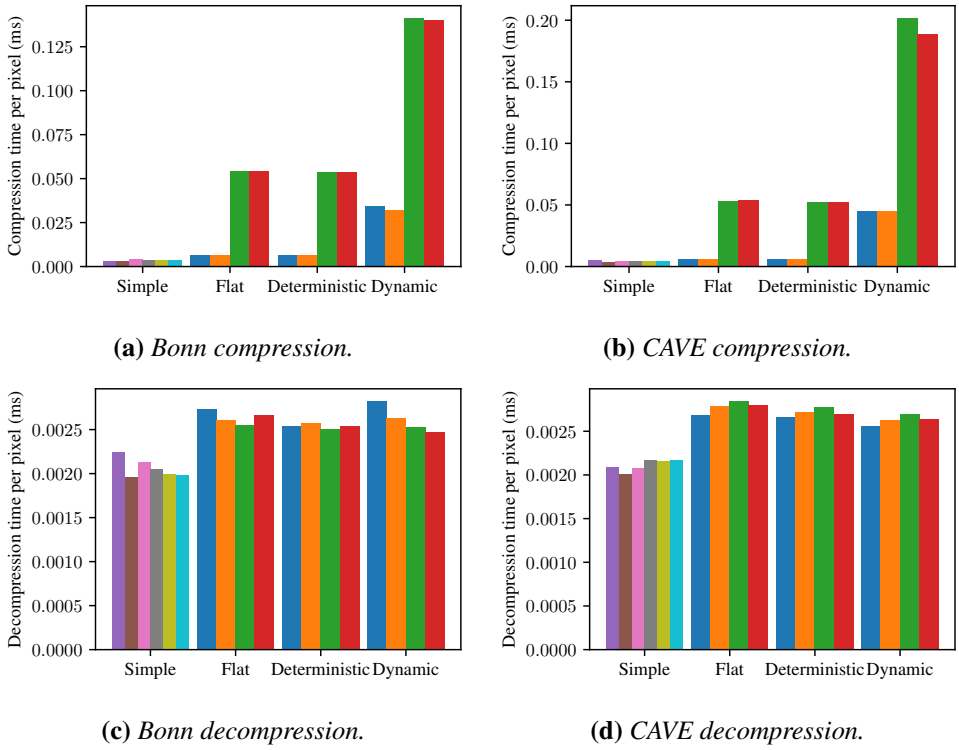


Figure 9. Statistics on errors and compression ratios for our two databases. Bar plots show average values across all images; scatter plots show results for each image individually. The legend lists parameter values for the various distance levels and the subsampling ratio.



(a) *Bonn compression.*

(b) *CAVE compression.*

(c) *Bonn decompression.*

(d) *CAVE decompression.*

Figure 10. Average computation times per pixel for our two databases.

component (red bars/markers). Even without chroma subsampling, our more aggressive configuration (orange bars/markers) achieves average compression ratios of 7.22 on the Bonn database and 29.02 on the CAVE database compared to OpenEXR 32-bit float framebuffers with ZIP compression.

4.5. Run Time

We measure timings on an Intel Core i7-13700K CPU and 32 GB of RAM with a Samsung SSD 990 PRO 2TB NVMe SSD. Figure 10 shows average timings, and the supplemental materials provide timings for each individual image. For compression, our method with flat or deterministic compression curves and no chroma subsampling is slower than the simple method, but compression only takes about twice as long as decompression. Chroma subsampling has a surprisingly high cost during compression, presumably due to the sophisticated implementation of downsampling in libjxl. The overhead for the optimization of compression curves is substantial. In many cases, it could be hard to justify because deterministic curves also work well. During decompression, the overhead of our method is low and timings for all techniques are similar.

5. Conclusion

Large file sizes are one of the major remaining drawbacks when transitioning from an RGB renderer to a spectral renderer. Our work is the first to address this problem directly but probably not the last. By separating mean brightness and chrominance, our format achieves good compression ratios. However, limitations of the latest libjxl keep us from storing complete spectral images in a single file, which makes it less convenient to handle our spectral JPEG XL. There is further potential for compression by exploiting redundancies across different channels more effectively. In production rendering, there may also be a strong desire to have lossless compression with better compression ratios compared to spectral OpenEXR. We hope that our work helps to put the problem of spectral image compression on the map and that it will motivate improvements to libjxl.

Desirable features for future versions of libjxl would be:

- Handling RGB framebuffers and scalar-valued framebuffers in the same file (for RGB previews),
- Supporting an arbitrary number of framebuffers (currently limited to 256),
- Independent control of distance levels and resolutions for different framebuffers within the same file.

Acknowledgements

We want to thank Sebastian Merzbach for providing spectral measurements of materials in the Bonn database. We also thank the JPEG XL developers for answering our questions.

References

ABOUSLEMAN, G. P., MARCELLIN, M. W., AND HUNT, B. R. Compression of hyperspectral imagery using the 3-D DCT and hybrid DPCM/DCT. *IEEE Transactions on Geoscience and Remote Sensing*, 33(1):26–34, 1995. URL: https://doi.org/10.1109/36.368225. 52

ENVI. ENVI image files. March 4th, 2025. URL: https://www.nv5geospatialsoftware.com/docs/enviimagefiles.html. 51

FICHET, A., PACANOWSKI, R., AND WILKIE, A. An OpenEXR layout for spectral images. *Journal of Computer Graphics Techniques (JCGT)*, 10(3):1–18, September 2021. URL: http://jcgt.org/published/0010/03/01/. 50, 51

ISO. Information technology—JPEG XL image coding system. Technical Report ISO/IEC 18181-1:2024, International Organization for Standardization, 2024. URL: https://www.iso.org/standard/85066.html. 52, 55

ITU. Digital compression and coding of continuous-tone still images—Requirements and guidelines. Technical Report T.81 (09/92), International Telecommunication Union, 1992. URL: https://www.itu.int/rec/T-REC-T.81-199209-I/en. 52

JAKOB, W. AND HANIKA, J. A low-dimensional function space for efficient spectral upsampling. *Computer Graphics Forum*, 38(2):147–155, March 2019. URL: https://doi.org/10.1111/cgf.13626. 52

KAARNA, A. Compression of spectral images. In OBINATA, G. AND DUTTA, A., editors, *Vision Systems*, pages 269–298. IntechOpen, 2007. URL: https://doi.org/10.5772/4964. 52

MERZBACH, S., HERMANN, M., RUMP, M., AND KLEIN, R. Learned fitting of spatially varying BRDFs. *Computer Graphics Forum*, 38(4):193–205, 2019. URL: https://doi.org/10.1111/cgf.13782. 60

OpenEXR. Reading and writing image files with the OpenEXR library. March 4th, 2025. URL: https://openexr.readthedocs.io/en/latest/ReadingAndWritingImageFiles.html. 51

PETERS, C., MERZBACH, S., HANIKA, J., AND DACHSBACHER, C. Using moments to represent bounded signals for spectral rendering. *ACM Transactions on Graphics*, 38(4), jul 2019. URL: https://doi.org/10.1145/3306346.3322964. 52, 53, 54, 60

RAPP, T., PETERS, C., AND DACHSBACHER, C. Image-based visualization of large volumetric data using moments. *IEEE Transactions on Visualization and Computer Graphics*, 28(6):2314–2325, 2022. URL: https://doi.org/10.1109/TVCG.2022.3165346. 56

SMITS, B. An RGB-to-spectrum conversion for reflectances. *Journal of Graphics Tools*, 4(4):11–22, 1999. URL: https://doi.org/10.1080/10867651.1999.10487511. 52

WILKIE, A., NAWAZ, S., DROSKE, M., WEIDLICH, A., AND HANIKA, J. Hero wavelength spectral sampling. *Computer Graphics Forum*, 33(4):123–131, 2014. URL: https://doi.org/10.1111/cgf.12419. 50

YASUMA, F., MITSUNAGA, T., ISO, D., AND NAYAR, S. K. Generalized assorted pixel camera: Postcapture control of resolution, dynamic range, and spectrum. *IEEE Transactions on Image Processing*, 19(9):2241–2253, 2010. URL: https://doi.org/10.1109/TIP.2010.2046811. 51, 59

Index of Supplemental Materials

We provide a sample implementation of the compressor and decompressor in C++, as well as a PDF with detailed results for all images in our two databases.

- Supplemental code:
  https://jcgt.org/published/0014/01/04/supplemental-code.zip
- Detailed results:
  https://jcgt.org/published/0014/01/04/supplemental.pdf

Author Contact Information

Alban Fichet
Intel Corporation
alban.fichet@gmx.fr
https://afichet.github.io

Christoph Peters
Intel Corporation
paper@momentsingraphics.de
https://momentsingraphics.de

Alban Fichet and Christoph Peters, Compression of Spectral Images Using Spectral JPEG XL, *Journal of Computer Graphics Techniques (JCGT)*, vol. 14, no. 1, 49–69, 2025
http://jcgt.org/published/0014/01/04/

Received:	2023-01-30		
Recommended:	2024-12-14	Corresponding Editor:	Brent Burley
Published:	2025-03-04	Editor-in-Chief:	Eric Haines