Storing of High Dynamic Range Images in JPEG/JFIF files

Konrad Kabaja*

Institute of Computer Graphics and Multimedia Systems
Szczecin University of Technology
Poland

Abstract

There's a plenty of file formats designed for storing high dynamic range (HDR) images. But none of them provide backwards compatibility with existing file formats. In this paper we propose a simple approach to lossy compressing HDR data in a standard JPEG/JFIF file. An original HDR file is decomposed into a tone mapped version of it LDR (low dynamic range) 24bpp and additional information needed to restore the original HDR image from LDR. The LDR image is compressed using standard JPEG encoder and forms a foreground image. Extra data is carried wavelet-compressed as a subband in the file. Naive (not HDR-enabled) applications will see the image as a normal JPEG/JFIF file and will extract only the LDR data giving user a preview of the original HDR image. HDR-enabled applications will extract additional information from the subband channel and restore the HDR image. The results of compressing a series of natural and synthetic HDR images using various lossy compression settings are presented and discussed in the paper.

Keywords: High dynamic range image formats, lossy compression, image processing, JPEG, wavelet compression.

1 Introduction

Visible light in real world covers very wide range of luminance. Humans can perceive 4 orders of magnitude simultaneously and can adapt their eyes to see 8 orders. Conventional digital image storing formats (24bpp) are capable of representing only 2 orders of magnitude and are called LDR low dynamic range, output referred images. The genesis of this limitation is because standard CRT/LCD displays are capable of displaying only 2 orders of luminance and a limited gamut of colors. The image formats that can hold extended gamut and dynamic range are called HDR high dynamic range, scene referred images. First attempts to develop HDR displays have been made by Seetzen et al. [10]. The methods for obtaining HDR images are already known and a variety of storing formats has been introduced.

The main issue is that all HDR image formats provide no backward compatibility, which slows the adaptation of HDR imagery into end-user solutions. We need a standard, compact, backward compatible and lossy format, which can be adapted for storing large HDR images on small storage space like JPEG. That format would be used for storing images taken by HDR enabled photo cameras without a fear that photos wouldn't be opened by naive applications. Such file would be seen by standard applications as plain JPEG/JFIFimage, offering a LDR version of the original HDR image. The file would be seen as the HDR image by HDR-enabled applications.

2 Background

First attempts to store high luminance values has been made in late '80 by researchers (e.g. Jourlin & Pinoli 1988 [4]). Then in 1989 Radiance system introduced RGBE format for storing HDR data. In 1998 LogLuv representation was proposed by Ward Larson [6]. Color pixels were encoded as log luminance values and CIE (u',v') chromaticity coordinates. In this format, encoding was implemented and distributed later as part of the standard TIFF I/O library. In 2002 Industrial Light and Magic made their EXR format available Kains et al. 2002 [5]. It supports 16-bit floating-point (supported natively on Nvidia's GeForce/Quadro FX GPU's & Cg graphics language), 32-bit floating-point, and 32-bit integer pixels. The format supports also lossless & lossy image compression algorithms. In 2004 Greg Ward & Maryann Simmons [12] proposed a simple approach to HDR encoding (in JPEG files), which was backward compatible. This paper is based on results achieved in that article.

The first attempts to effectively compress HDR animations in MPEG format were made Rafal Mantiuk et al. 2004 [9]. However, compression of animation will not be discussed in this paper.

3 Method description

3.1 Coding/decoding pipeline

Figure 1 shows an overview of HDR-JPEG encoding pipeline. We start with the HDR scene-referred image. It is then processed by tone mapping operator (TMO) that gives in result output-referred LDR image. This method

*kkabaja@wi.ps.pl

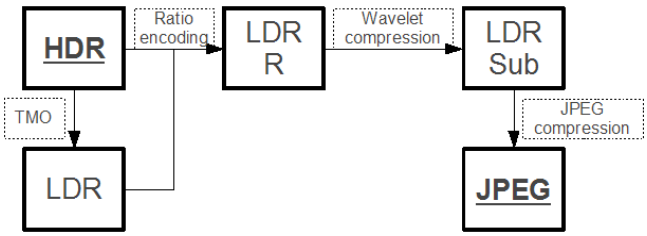

Figure 1: HDR-JPEG encoding pipeline.

was invented to work with any TMO, but in our pipeline we use Drago et al. TMO [3]. In our opinion, it gives the best visual results for LDR preview that user will see in a naive application. In the next step HDR and LDR images are encoded giving a composite, consisting of previous LDR image (not modified) and a ratio image R that holds information needed for restoring HDR image from LDR image. Later on, wavelet compression is applied to R channel giving compressed subband data (Sub). This method was invented to work with any wavelet encoder. In our research we've used SFQ by Xiong et al. 1996 [13] and TCE by Tian and Hemami 2004 [11] encoders that present the lowest distortion ratios. In the last step, LDR image is compressed using a standard JPEG encoder and subband data are attached to JFIF header giving in result a backward compatible JPEG/JFIF file containing the additional data.

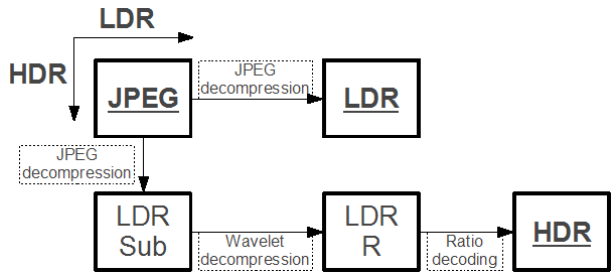

Figure 2: HDR-JPEG decoding pipeline.

As we can see in Figure 2 there are two ways that HDR-JPEG file can be treated. Naive applications will utilize only the simpler method (upper path) consisting of only JPEG-decompression step that will extract only output-referred LDR image data. This path provides backward compatibility for applications that don't care about additional HDR information stored in the file. HDR enabled applications will choose the lower path. First, JFIF header will be unpacked and JPEG data decompressed resulting in an LDR image and extra data Sub. Next, subband will be decompressed using wavelet decompressor giving in result ratio image R. The last step restores an HDR scene-referred image from the LDR image and R ratio image.

3.2 Ratio image

The variety of tone mapping operators that can be used is limited by two rules that have to be followed by them:

- The original HDR image must be smoothly mapped to a 24-bit output domain.
- Hue must be maintained at each pixel.

Restricted by these rules, we can compute ratio image R by dividing HDR luminance by LDR luminance and applying some additional computations to minimize errors provided by lossy compression and integer representation:

$$R = \frac{\mathrm{Luma}(HDR)}{\mathrm{Luma}(LDR)}. \quad (1)$$

Ratio R data will be stored in data markers reserved for additional data in JPEG format. JFIF header offers us a limited set of additional data markers that can be added to file. These are 16 Application Markers (APP0 to APP15) and one User Comment Marker (COM). Every marker is limited to 64Kbytes of storage space, because it's length is stored on 2 bytes in header. APPx markers are allowed to hold arbitrary binary or text data. APP0 marker is reserved for standard JFIF v1.2 header and APP14 is reserved by Adobe for theirs additional data. APPx markers are perfect suitable for keeping our subband. We will utilize 14 from 16 available markers resulting in 895 Kbytes of additional binary space for our Sub data. The COM marker can also contain arbitrary binary or text data because it is treated the same way in JFIF header as the APPx markers. Its size is also limited to 64Kbytes. However, the purpose of COM marker is to hold plain-text-only data, which user can add manually and, following the JFIF specification, it is highly undesirable to hold binary data here. But this doesn't mean that COM marker is useless for us. We will hold there some additional data needed for the decompression of HDR image (to be specific - two floating-point range factors stored in text format).

Figure 3: LDR foreground image (left) and ratio image (right).

3.3 Compression of ratio image

Because the ratio image may be very large for very large HDR images, and we have only limited storage space we have to compress it. We could use JPEG compressor to compress the ratio image, but we've chosen wavelet compression because it gives better compression ratios at the same quality. Because of the JPEG storage limitation, there may be situation when we exceed this limit. In this situation Ward et al. [12] proposes downsampling the ratio image. This method has a drawback resulting in drastically lowering the amount of information kept in ratio image. Ward introduces pre- or postprocessing step that fixes these lacks by modifying the LDR image. We propose lowering bitrate quality settings of the wavelet compressor and repeating the compression step to decrease the size of the subband. This eliminates the need for an additional pre- or postcorrection step.

However, it doesn't mean that postcorrection step may not be introduced. It could consist of decompressing, compressed subband (ratio image) data and correcting LDR image to eliminate errors introduced by wavelet compression (this technique would be similar to one described by Ward et al. [12] for downsampled images).

4 Method details

4.1 Compression

Step 1. Apply a TMO to a HDR image stored in XYZ colorspace. In this step the following formula will be applied to XYZ:

$$\begin{bmatrix} tX_{ij} \\ tY_{ij} \\ tZ_ij \end{bmatrix} = \begin{bmatrix} X_{ij} \\ Y_{ij} \\ Z_{ij} \end{bmatrix} \frac{L_{ij}}{Y_{ij}}. \tag{2}$$

Where L_{ij} is a luma channel computed by TMO. It will give in result $tXYZ_{ij}$. Note that tY_{ij} is in effect a L_{ij} channel returned by TMO. Values of $tXYZ_{ij}$ range from 0 to 1.

Step 2. Transform to the RGB colorspace. Values of $tRGB_{ij}$ range from 0 to 1, similar to $tXYZ_{ij}$. The matrix that we use is a standard XYZ2RGB transformation matrix with D65 white point.

$$\begin{bmatrix} tR_{ij} \\ tG_{ij} \\ tB_ij \end{bmatrix} = \begin{bmatrix} 3.2406 & -1.5372 & -0.4986 \\ -0.9689 & 1.8758 & 0.0415 \\ 0.0557 & -0.2040 & 1.0570 \end{bmatrix} \begin{bmatrix} tX_{ij} \\ tY_{ij} \\ tZ_ij \end{bmatrix} \tag{3}$$

Step 3. Because JPEG compressor takes integer values as an input we have to scale floating-point range $\langle 0, 1\rangle$ of $tRGB_{ij}$ to $\langle 0, 255\rangle$ and quantize it to integer values $\langle 0, 255\rangle$. This represents a LDR image that will be compressed using JPEG later. Last step is to scale it back to $\langle 0, 1\rangle$ $itRGB_{ij}$ because we will need it in the later computations.

$$\begin{bmatrix} itR_{ij} \\ itG_{ij} \\ itB_{ij} \end{bmatrix} = \frac{\text{Integer}(255 \cdot \begin{bmatrix} tR_{ij} \\ tG_{ij} \\ tB_ij \end{bmatrix})}{255} \tag{4}$$

Step 4. Transform back to XYZ colorspace. Values of $itXYZ_{ij}$ range from 0 to 1, but are quantized to only 256 different values. The matrix used is a standard RGB2XYZ transformation matrix with D65 white point.

$$\begin{bmatrix} itX_{ij} \\ itY_{ij} \\ itZ_{ij} \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} itR_{ij} \\ itG_{ij} \\ itB_{ij} \end{bmatrix} \tag{5}$$

Step 5. Divide Y_{ij} by $(itY_{ij} + 1)$, the result is R our ratio image.

$$R_{ij} = \frac{Y_{ij}}{itY_{ij} + 1} \tag{6}$$

Why does we need Step 2-4 and why we choose itY_{ij} instead of tY_{ij} in equation? Because JPEG needs integer values and in effect it compresses $itRGB_{ij}$ not $tXYZ_{ij}$. So, after decompression we will obtain $itRGB_{ij}$ not $tXYZ_{ij}$ or $tRGB_{ij}$ and some information will be lost. Choosing tY_{ij} would introduce a small additional error in the decompression stage.

Why does we add 1 to itY_{ij}? Because itY_{ij} ranges from 0 to 1 and there would be many divide by zero errors. Even if itY_{ij} would be very close to 0 (but not equal 0) it would introduce rounding errors during computations in CPU/FPU. To avoid this, we translate $\langle 0, 1\rangle$ range to $\langle 1, 2\rangle$.

Step 6. Apply a decimal logarithm to ratio image R. This step compresses a linear R to non-linear lR. We introduced this step because great number of images utilizes low luminance values more frequently than high values. In effect, we assign greater bandwidth in subband channel for darker luminance values.

$$lR_{ij} = \log_{10}(R_{ij}) \tag{7}$$

Step 7. Because after Step 6 lR_{ij} can carry large and negative values we have to scale it to a static range $\langle 0 - 255\rangle$. We will also need delta and minim values for decompression. These values will be remembered in output image file.

$$\begin{aligned} minim &= \min(lR) \\ delta &= \max(lR) - \min(lR) \\ slR_{ij} &= 255(\frac{lR_{ij} - minim}{delta}) \end{aligned} \tag{8}$$

Step 8. Quantize slR_{ij} to integer values before compression. This is our final ratio image. That we will use during decompression to restore HDR data.

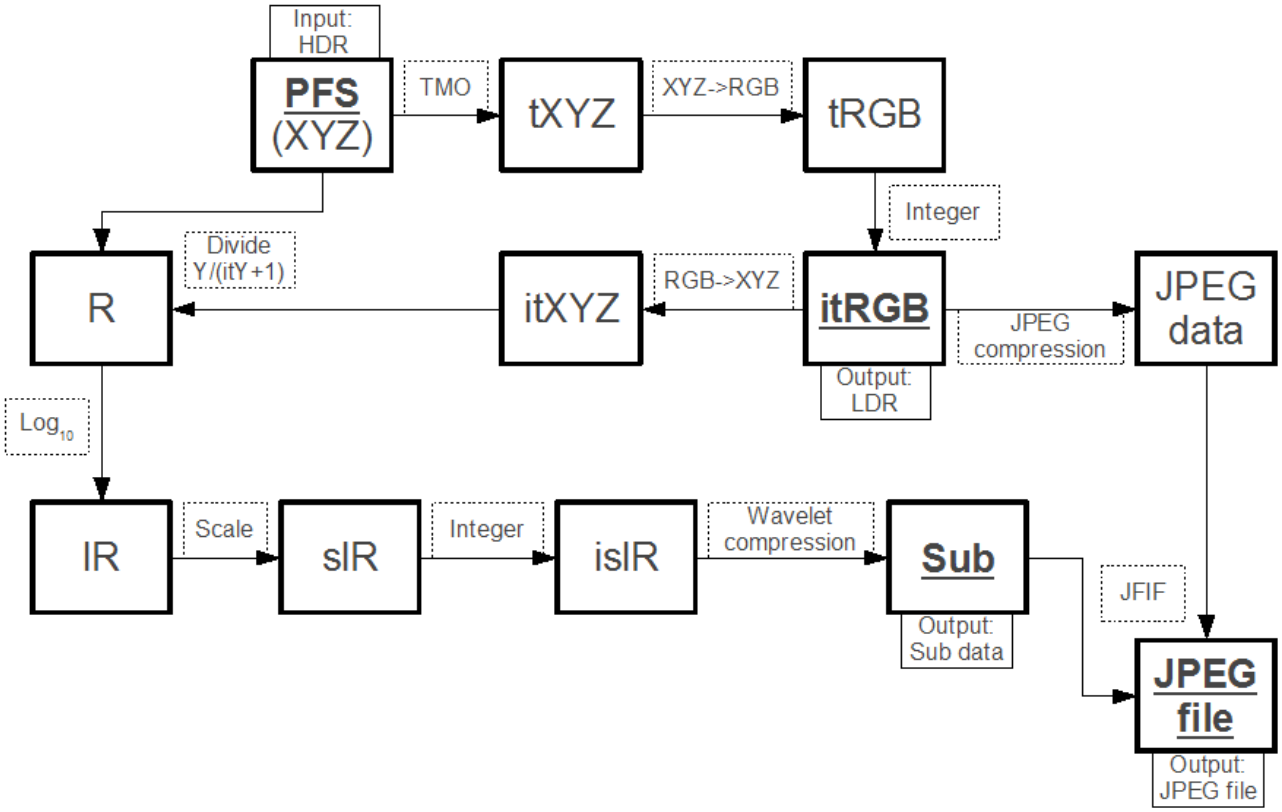

Figure 4: HDR-JPEG image encoding pipeline (details).

$$islR_{ij} = \text{Integer}(slR_{ij}) \tag{9}$$

Step 9. Compress $islR$ using wavelets. If compressed data extends 895 Kbytes data limit this step should be repeated with lower bitrate settings for wavelet compressor as many times as needed. Because this step may be time consuming, it should be implemented very careful (especially the bitrate-choosing algorithm) to minimize the needed repetitions. If chosen wavelet compressor supports specifying target bitrate a priori then it should be set to the proper value to eliminate repetitions at all.

In proposed method we don't scale down image (as Ward et al. [12] does) because ˜900 Kbytes subband limit provides plenty of storage data space for great amount of processed images. We don't need post-correction step on LDR image to correct lost information in subband.

Why does we need Step6-8? Because the wavelet compressor takes only 8-bit integer values as an input and cannot store full 32-bit information in floating-point format. We have to do everything to utilize this small amount of data space the best way.

$$Subdata = \text{WaveletCompress}(islR) \tag{10}$$

Step 10. Compress LDR output-referred image $itRGB$ using JPEG. This is a standard JPEG compression.

$$JPEGdata = \text{JPEGCompress}(itRGB) \tag{11}$$

Step 11. Now we have everything that we need to create HDR-JPEG file. Finally we pack subband data in JFIF header and add $JPEGdata$ to form a proper JPEG/JFIF file. Sub data are divided into up to 14 64Kbytes-packets and stored in APPx markers. We also store minim and delta values needed to restore the ratio image in COM marker in text form. We are also able to provide additional information in COM marker, e.g. used wavelet compressor, information about image content, copyright, hardware used to take photo etc.

$$JPEGfile = \text{JFIFPack}(Subdata, JPEGdata) \tag{12}$$

$JPEGfile$ is the final output file that can be viewed using naive application. It also contains additional subband information that can be accessed via HDR enabled application to restore an HDR image.

4.2 Decompression

Step 1. JPEG/JFIF file is unpacked to compressed JPEG-data and Sub data (if the application utilizes additional subband information). Sub data is compacted from up to 14 64 Kbytes APPx markers. Also the minim and delta

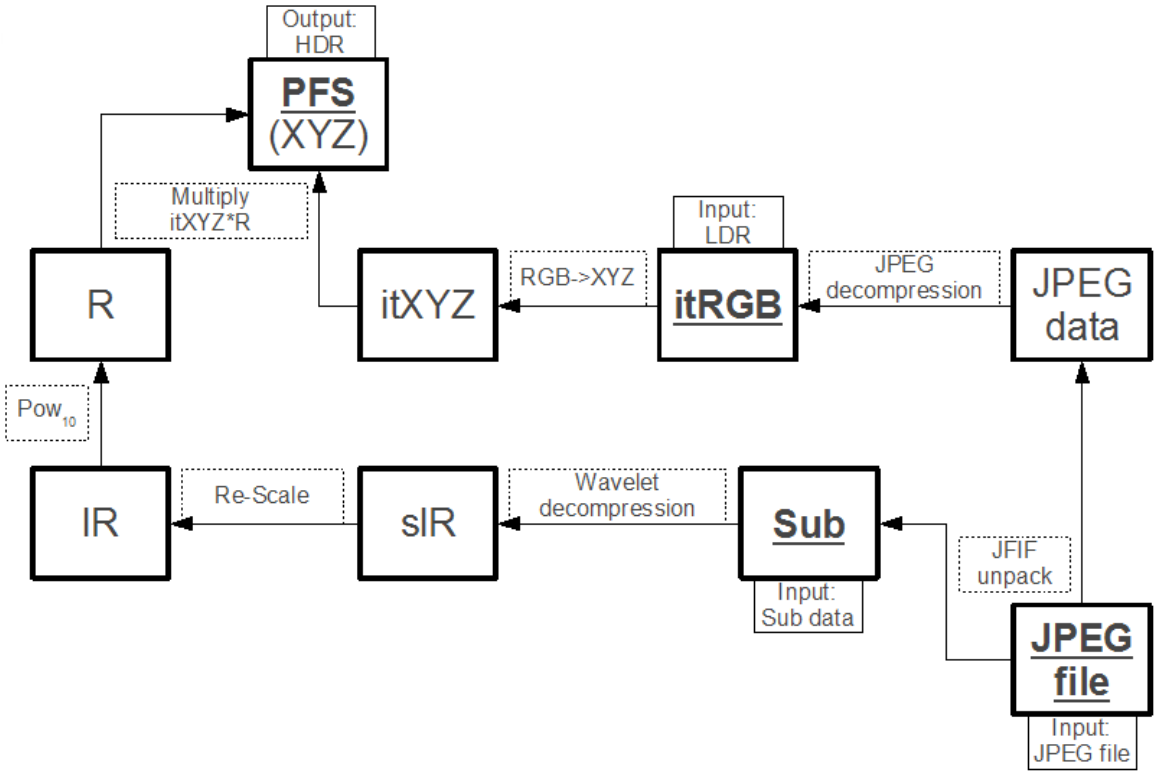

Figure 5: HDR-JPEG image decoding pipeline (details).

factors (mentioned in Section 3.4.1) are restored from COM marker.

$$Subdata, JPEGdata = \text{JFIFUnpack}(JPEGfile) \tag{13}$$

Step 2. $JPEGdata$ is uncompressed to LDR output-referred image. This step is the last one for naive applications , which ignore additional subband information in APPx markers. Naive application:

$$itRGB = \text{JPEGDecompress}(JPEGdata). \tag{14}$$

HDR enabled application:

$$itRGB = \frac{\text{JPEGDecompress}(JPEGdata)}{255}. \tag{15}$$

For HDR applications $itRGB$ is scaled to range $\langle 0, 1\rangle$.

Step 3. Switch to XYZ colorspace. Values of $itXYZ_{ij}$ range from 0 to 1. The matrix used is a standard RGB2XYZ transformation matrix with D65 white point.

$$\begin{bmatrix} itX_{ij} \\ itY_ij \\ itZ_ij \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} itR_{ij} \\ itG_{ij} \\ itB_{ij} \end{bmatrix} \tag{16}$$

Step 4. Uncompress ratio image from subband data.

$$slR = \text{WaveletDecompress}(Subdata) \tag{17}$$

Step 5. Re-Scale ratio image to its original space using delta and minim factors.

$$lR_{ij} = \frac{slR_{ij}}{255} \cdot delta + minim \tag{18}$$

Step 6. Uncompress non-linear ratio image representation to linear.

$$R_{ij} = 10^{lR_{ij}} \tag{19}$$

Step 7. Finally restore HDR scene-referred image from LDR output-referred image and ratio image.

$$\begin{aligned} Y_{ij} &= (itY_{ij} + 1) \cdot R \\ X_{ij} &= itX_{ij} \cdot \frac{(itY_{ij} + 1) \cdot R_{ij}}{itY_{ij}} \\ Z_{ij} &= itZ_{ij} \cdot \frac{(itY_{ij} + 1) \cdot R_{ij}}{itZ_{ij}} \end{aligned} \tag{20}$$

How does it work? Rij is equal to (see previous section):

$$R_{ij} = \frac{Y_{ij}}{itY_{ij} + 1}, \tag{21}$$

so we can substitute it:

$$
\begin{aligned}
Y_{ij} &= (itY_{ij} + 1) \cdot R_{ij} \\
&= (itY_{ij} + 1) \cdot (\frac{Y_{ij}}{itY_{ij} + 1}) \\
&= Y_i j.
\end{aligned}
\tag{22}
$$

Also we remember that $itY_{ij} = L_{ij}$ and itX_{ij} was computed in compression step as:

$$
\begin{aligned}
itX_{ij} = tX_{ij} = X_{ij} \cdot \frac{L_{ij}}{Y_{ij}}, \\
itY_{ij} = L_{ij}.
\end{aligned}
\tag{23}
$$

We approximate tX_{ij} using itX_{ij} because we lost this detailed information in compression step. It will introduce small error in the output HDR image.

We can substitute:

$$
\begin{aligned}
X_{ij} = itX_{ij} \cdot \frac{(itY_{ij} + 1) \cdot R_{ij}}{itY_{ij}}, \\
X_{ij} = X_{ij} \cdot (\frac{itY_{ij}}{Y_{ij}}) \cdot \frac{(itY_{ij} + 1) \cdot (\frac{Y_{ij}}{itY_{ij}+1})}{itY_{ij}} = X_{ij}.
\end{aligned}
\tag{24}
$$

Z value computations are similar to X.

5 Implementation

HDR-JPEG encoding and decoding programs (called pfsinjpeg and pfsoutjpeg) were implemented as additional parts to PfsTools package [8]. For tone mapping operators implementation PfsTMO package was used [7]. We used JPEG library provided by Independend JPEG Group [1] to compress and decompress LDR data. QccPack library [2] was utilized for wavelet compression and decompression of subband data.

6 Tests

5 natural, realistic and 2 synthetic, rendered images were chosen for tests (Figure 6-9). They range from small to very large and represent various content. Dynamic range varies from 2.8 to 4.8 (e.g. 4.8 is $1 : 10^{4.8}$ or $1 : 63,000$ dynamic range). The image characteristics were shown in Table 1. The following quality settings were used for compression:

- LDR data encoded using JPEG quality 70 (max 100) and subband data encoded using SFQ algorithm by Xiong et al. [13]. SFQ settings (standard): wavelet CohenDaubechiesFeauveau.9-7; boundary extension symmetric; rate-distortion parameter 1.0; baseband and highpass quantizers were allowed to search all allowable stepsizes; number of levels of dyadic decomposition 3.

Figure 6: Display1000 and Montreal images.

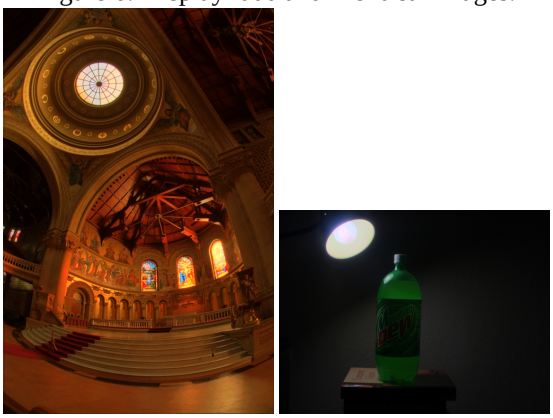

Figure 7: Memorial and Mountain Dew images.

Figure 8: Rend09 and Rend13 images.

Figure 9: Price Western image.

- LDR data encoded using JPEG quality 100, subband data encoded using SFQ algorithm with the same values as in previous step.

- LDR data encoded using JPEG quality 100, subband data encoded using TCE algorithm by Tian and Hemami [11]. TCE settings (standard): wavelet CohenDaubechiesFeauveau.9-7; boundary extension symmetric; number of levels of dyadic decomposition 5; target bitrate 2.

We've chosen Drago et al. [3] tone mapping operator for the tests. Target file and subband data sizes were measured

and compression ratio factors were computed. The resulting JPEG images were decompressed to HDR images and compared to HDR originals using MSE (mean square error) for each X, Y, Z channel and an average XYZ MSE.

7 Results

Tables 3 and 4 show the achieved file sizes (also subband data sizes) and mean square errors measured between original HDR image and decompressed HDR-JPEG image. For the first series of tests (Q70 + SFQ) the achieved compression ratios (CR, Table 2.) were impressive. Lowest CR was 11.8 for ultra-large Price Western and the highest was for Mountain Dew with 38.1 (average 24.5). The cost for such big CR was quite large quality drop. Lot of artifacts were visible and images lacked details. We have observed quite interesting pecularity after decompressing highly compressed HDR-JPEG image to HDR and applying a TMO on it again. The final image presented far better quality than LDR image in original HDR-JPEG. We present this situation in Figure 10.

The explanation for this behaviour is the fact that (in this quality setting) almost half of the file is occupied by compressed ratio image. Subband data is compressed approximately 7-12:1 having two to three times more bandwidth space than LDR foreground image (color + luminance!). After decompression, overall quality is improved by additional (less compressed) data from ratio image.

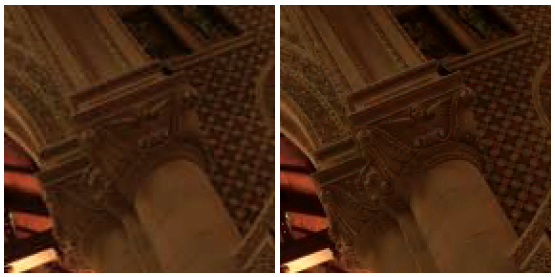

Figure 10: JPEG image (left), decompressed and tone mapped image (right). Right image has more details than left one.

Next tests (Q100 + SFQ) present acceptable results. We noticed lost of details in smooth areas. Images were noisy in high detail places, but overall quality was quite good. Compression ratios varied from 3.6 (Memorial) to 9.2 (Mountain Dew) with average 5.9 4 times more data compared to the first test.

The last series of tests show really good image quality. Images are clean, without artefacts, lost of details or noise. We have increased bitrate for wavelet compressor to 2bpp so, subband was compressed at 4:1 ratio. The cost of good quality are quite large files and low compression ratios. The average is 4.3, but it is still far better than OpenEXR's lossless PIZ compression with average CR = 1.6 Kains et al. 2002 [5]. The test for Price Western image failed and results weren't computed. That was because at this quality setting there was too little space in application markers to store subband data. We could lower bitrate settings, but we left it unchanged to signalize the problem.

We noticed some common issues that appeared at all compression settings. The first issue is the lack of details in very bright areas (Figure 11).

Figure 11: Original tone mapped image (top left), JPEG foreground image (top right), difference between images (bottom).

This is caused by logarithmic compression of luminance. We assigned wider bandwidth in ratio image for low values to expose darker areas, but the drawback is less space for high values. The second issue is the noticeable quantization of values (that was done for JPEG and wavelet input as well because compressors needed integer values). It revealed itself in smooth areas e.g. sky, which looks like it would have 16bpp palette. Another drawback of this method is the compression time. Compression of Display1000 2048x1536 10Mbytes image takes about 6 minutes on P4 2.0GHz using SFQ algorithm. For TCE it takes about 30 seconds. For example JPEG needs only 5 seconds to compress this image (but only LDR part).

Table 1: Testing images.

Name	Resolution	Dynamic range	Source
Display1000	2048x1536	3.4	n/a
Memorial	512x768	4.8	Paul Debevec
Montreal	2048x1536	3.1	n/a
Mountain Dew	2048x1536	n/a	n/a
Price Western	3272x1280	2.8	Spheron
rend09	1024x1024	3.9	Saba Rofchaei
rend13	1024x1024	4.1	n/a

8 Conclusions and future work

We proposed an algorithm for lossy, high dynamic range image compression format , which is backward compati-

Table 2: Resulting compression ratios.

Name	CR, JPEG Q70, SFQ	CR, JPEG Q100, SFQ	CR, JPEG Q100, TCE
Display1000	17.6:1	4.4:1	3.6:1
Memorial	12:1	3.6:1	3.2:1
Montreal	31.9:1	5.8:1	4.2:1
Mountain Dew	38.2:1	9.2:1	5.8:1
Price Western	11.8:1	3.6:1	n/a
rend09	30.3:1	7.3:1	4.8:1
rend13	29.7:1	8:1	4.2:1
Average:	24.5:1	5.9	4.3

Table 3: Resulting file sizes.

Name	HDR size	JPEG size, Q70, SFQ + subdata size	JPEG size, Q100, SFQ + subdata size	JPEG size, Q100, TCE + subdata size
Display1000	10 MB	583 KB 244 KB	2.27 MB 244 KB	2.78 MB 768 KB
Memorial	1.28 MB	109 KB 56 KB	366 KB 56 KB	406 KB 96 KB
Montreal	9.45 MB	303 KB 144 KB	1.62 MB 144 KB	2.23 MB 768 KB
Mountain Dew	9.57 MB	256 KB 156 KB	1.04 MB 156 KB	1.64 MB 768 KB
Price Western	13.4 MB	1.14 MB 674 KB	3.65 MB 674 KB	n/a
rend09	2.93 MB	99 KB 53 KB	412 KB 53 KB	624 KB 256 KB
rend13	1.92 MB	66 KB 36 KB	244 KB 36 KB	463 KB 256 KB

ble. It is one of the first lossy compression formats proposed in this matter. The main advance of the proposed solution is that it utilizes the existing JPEG compression standard for data storage. Images stored using this method can be read in every hardware or software that is capable of reading standard JPEG files. In HDR-enabled applications or devices it can be read as HDR image. This fills the gap between HDR imaging technology and existing LDR solutions.

We see some deficiencies in proposed method, which point out future work. Logarithmic luminance compression should be replaced by some more intelligent solution that would compress it in an adaptive way. Wavelet compression methods should be reviewed to choose the best suitable for the HDR image compression. A method for minimizing integer numbers round-off errors should be developed. A postcorrection step should be considered to improve image quality.

9 Acknowledgments

Thanks to Radoslaw Mantiuk for providing lots of help during writing process of this article. Thanks to Rafal Mantiuk for the first concept of this algorithm and PfsTools package.

Table 4: Resulting MSE.

Name	JPEG Q70, SFQ MSE	JPEG Q100, SFQ MSE	JPEG Q100, TCE MSE
Display1000	X: 0.034898 Y: 0.028088 Z: 0.074218 XYZ: 0.045735	X: 0.034147 Y: 0.027747 Z: 0.061485 XYZ: 0.041126	X: 0.013952 Y: 0.005275 Z: 0.042642 XYZ: 0.020623
Memorial	X: 0.222154 Y: 0.136199 Z: 0.086948 XYZ: 0.148433	X: 0.217203 Y: 0.132986 Z: 0.067651 XYZ: 0.139280	X: 0.144987 Y: 0.048072 Z: 0.025714 XYZ: 0.072924
Montreal	X: 0.009108 Y: 0.002706 Z: 0.010072 XYZ: 0.007295	X: 0.008893 Y: 0.002578 Z: 0.007879 XYZ: 0.006450	X: 0.007818 Y: 0.001364 Z: 0.006735 XYZ: 0.005306
Mountain Dew	X: 0.000001 Y: 0.000001 Z: 0.000007 XYZ: 0.000003	X: 0.000001 Y: 0.000001 Z: 0.000007 XYZ: 0.000003	X: 0.000001 Y: 0.000000 Z: 0.000007 XYZ: 0.000003
Price Western	X: 0.003506 Y: 0.003299 Z: 0.009585 XYZ: 0.005463	X: 0.003234 Y: 0.003134 Z: 0.008443 XYZ: 0.004937	n/a
rend09	X: 0.003630 Y: 0.003986 Z: 0.005297 XYZ: 0.004304	X: 0.003419 Y: 0.003776 Z: 0.004579 XYZ: 0.003925	X: 0.000158 Y: 0.000171 Z: 0.000296 XYZ: 0.000208
rend13	X: 0.005119 Y: 0.005685 Z: 0.006896 XYZ: 0.005900	X: 0.005027 Y: 0.005565 Z: 0.006614 XYZ: 0.005735	X: 0.001358 Y: 0.001503 Z: 0.001801 XYZ: 0.001554

References

[1] Independent jpeg group, http://ijg.org/.

[2] Qccpack library, http://qccpack.sourceforge.net/.

[3] F. Drago, K. Myszkowski, T. Annen, and N. Chiba. Adaptive logarithmic mapping for displaying high contrast scenes. *Computer Graphics Forum*, 22(3):419–426, September 2003.

[4] M. Jourlin and J-C. Pinoli. A model for logarithmic image processing. *Journal of Microscopy*, 149(1):21–35, 1998.

[5] F. Kains, R. Bogart, D. Hess, P. Schneider, and B. Anderson. Openexr, http://www.openexr.org/, 2002.

[6] Gregory Ward Larson. LogLuv encoding for full-gamut, high-dynamic range images. *Journal of Graphics Tools: JGT*, 3(1):15–31, 1998.

[7] Rafal Mantiuk and Grzegorz Krawczyk. Pfstmo, http://www.mpi-sb.mpg.de/resources/tmo/, 2004.

[8] Rafal Mantiuk and Grzegorz Krawczyk. Pfstools, http://www.mpi-sb.mpg.de/resources/pfstools/, 2004.

[9] Rafal Mantiuk, Grzegorz Krawczyk, Karol Myszkowski, and Hans-Peter Seidel. Perception-motivated high dynamic range video encoding. *ACM Transactions on Graphics*, 23(3):733–741, August 2004.

[10] Helge Seetzen, Wolfgang Heidrich, Wolfgang Stuerzlinger, Greg Ward, Lorne Whitehead, Matthew Trentacoste, Abhijeet Ghosh, and Andrejs Vorozcovs. High dynamic range display systems. *ACM Transactions on Graphics*, 23(3):760–768, August 2004.

[11] C. Tian and S. Hemami. An embedded image coding system based on tarp filter with classification. 2004.

[12] G. Ward and M. Simmons. Subband encoding of high dynamic range imagery. 2003.

[13] Zixiang Xiong, Kannan Ramchandran, and Michael T. Orchard. Space-frequency quantization for wavelet image coding. *IEEE Transactions on Image Processing*, 6(5):677–693, May 1997.