HIGH DYNAMIC RANGE IMAGE COMPRESSION ON COMMODITY HARDWARE FOR REAL-TIME MAPPING APPLICATIONS

Dirk Frommholz*, Daniel Hein, Marius Bock

DLR Institute of Optical Sensor Systems, Berlin, Germany - (dirk.frommholz, daniel.hein, marius.bock)@dlr.de

KEY WORDS: Image Compression, High Dynamic Range, Real-Time Mapping, Genetic Algorithm, Unmanned Aircraft System.

ABSTRACT:

This paper describes a lossy compression scheme for high dynamic range graylevel and color imagery for data transmission purposes in real-time mapping scenarios. The five stages of the implemented non-standard transform coder are written in portable C++ code and do not require specialized hardware to run. Storage space occupied by the bitmaps is reduced via a color space change, 2D integer discrete cosine transform (DCT) approximation, coefficient quantization, two-size run-length encoding and dictionary matching hinged on the LZ4 algorithm. Quantization matrices to eliminate insignificant DCT coefficients are derived from a representative image set through genetic optimization. The underlying fitness function incorporates the obtained output size, classic image quality metrics and the unique color count. Together with a zone-based adaptation mechanism, this allows to specify target bitrates instead of percentage values or abstract quality factors for the reduction rate to be directly matched to the available communication channel capacities. Results on a camera control unit of a fixed-wing unmanned aircraft system built around entry-level PC hardware revealed single-thread compression and decompression throughputs of several hundred mebibytes per second for full-swing 16 and 32 bit RGB imagery at medium compression ratios. A degradation in image quality compared to popular compression libraries could be identified, however, at acceptable levels statistically and visually.

1. INTRODUCTION

With ongoing advances in camera sensor technology and near-sensor pixel postprocessing, optical imagery with radiometric resolutions of fourteen, sixteen and even more bits per sample has started to make its way into various remote sensing applications. When the advantages of high dynamic range (HDR) bitmaps, i.e., capturing details under varying light conditions, are to be harnessed for airborne real-time mapping, the problem of high data volumes to be transmitted over communication channels of limited capacity within a narrow time frame arises. This applies to both the on-board image projection onto an existing surface model where ready-to-use map tiles have to be sent and to ground-based projection setups where raw or preprocessed image frames must be communicated.

In order to reduce the amount of data to match the capabilities of the available downlink, the images frequently get tone-mapped prior to compression which inevitably results in substantially less color nuances. Depending on the degree of image degradation, the information loss may not be desirable or acceptable at all. This applies in particular when operational pictures for natural or man-made disasters such as earthquakes, avalanches or major fires must be prepared without delay so that local rescue teams promptly can be given instructions to save human lives or prevent substantial economic damage. Here, the dynamic range of the imagery to be mapped has to be preserved as much as possible in order to recognize relevant scene details of the unknown surroundings on-site and under changing or poor illumination conditions. Such conditions especially occur during prolonged acquisition periods or when the power grid infrastructure has been impacted.

2. RELATED WORK AND MOTIVATION

Currently, there does not seem to be many publications that specifically deal with HDR image compression in a remote sensing context. In (Belyaev et al., 2017), 16 bit graylevel infrared images are subdivided into two low dynamic range (LDR) rasters. The LDR bitmap pairs to be compressed into standard JPEG (Pennebaker and Mitchell, 1992) or PNG (ISO, 2004) streams are obtained from the least and most significant bytes of the pixels. To control the degree of data reduction, discrete bitrate-distortion curves are computed from a set of test images. These curves subsequently get translated into the abstract quality factors expected by both image encoders. The split technique is once more applied by (Mantel and Forchhammer, 2017) to infrared bitmaps using the JPEG-XT still image processor (ISO, 2020b). In addition, the authors evaluate JPEG 2000 (ISO, 2019) and the MPEG-H Part 2/High Efficiency Video Coding (HEVC) (ISO, 2020a) extension on still image compression that directly work with 16 bit samples. Both approaches are purely software-based. Although no runtime figures are given for encoding and decoding, a substantial overhead can be expected at least when LDR bitmap pairs are to be dealt with.

Regarding real-time compression, (Manthey, 2014) describes a CCSDS 122.0-B-2-compliant architecture (CCSDS, 2017) to reduce the data volume of 16 bit satellite imagery on-board. It is based on a three-level 2D discrete wavelet transform (DWT) similar to JPEG 2000 with subsequent coefficient rearrangement and bitplane encoding. The incremental algorithm builds on field-programmable gate array (FPGA) hardware to achieve encoding speeds of around 400 MiB/s independently of the image content. It operates in either lossless or near-lossless mode to preserve the original content albeit this restricts the achievable compression ratios. The work of (Melián et al., 2021) embraces real-time transmission of hyperspectral data cubes of 12 bits per sample packed into a 16 bit unsigned type. It utilizes

* Corresponding author

a reduction algorithm named HyperLCA that was specifically tailored to this kind of imagery. HyperLCA comes along with a degradation in image quality (i.e., it is lossy) to achieve compression ratios of about 1:20. However, the encoder as a component of an unmanned aircraft system (UAS) relies on graphics processor hardware to handle the amount of roughly 87.5 MiB/s of raw data delivered by the camera module.

This paper proposes a custom compression algorithm which has been designed for the data transmission phase in HDR real-time mapping scenarios. Unlike in the previous work cited above, the described lossy encoder-decoder combination (codec) supports unsigned graylevel and RGB multispectral input bitmaps with radiometric resolutions between 12 and 41 bits per sample. This is beyond the capabilities of many of the established JPEG and MPEG standards. To achieve adequate compression ratios, bitmap storage is reduced through a sequence comprising a color space change, frequency domain transformation, zonal quantization of the resulting coefficients, run-length encoding and dictionary matching to be reversed for decompression. Image quality and compression ratios are controlled via target bitrates. The bitrate goals are given as bits per pixel values to be directly adapted to the data rate of the downlink communication channel. The individual stages of the algorithm have been selected, implemented as a portable C++ library and tuned for speed so they could run on inexpensive commodity hardware instead of graphics processors or FPGAs. Yet, it will be demonstrated that both the compressor and decompressor achieve single-thread performances sufficient to process 50 megapixel 16 bit RGB images of roughly 300 MiB per frame to a fraction of their original memory footprint and vice versa in much less than a second on low-end PC-like hardware that is part of an unmanned aircraft system.

3. COMPRESSION ALGORITHM DESCRIPTION

The proposed compression algorithm obeys the transform coder structure that is also followed by the ubiquitous JPEG codec. Image encoding starts with a colorspace transformation to separate luminance from chrominance when available. Due to the biology of the human visual system, color information can be condensed much stronger without noticeable artifacts. The luminance and chrominance channels afterwards get transformed into the frequency domain to decorrelate slow and fast intensity changes. Insignificant frequency components are set to zero by subsequent quantization. The quantization stage is followed by two lossless compression steps to reversibly condense the redundancies present in the stream of coefficients. While the two-size run-length encoder (RLE) shrinks consecutive sequences of the same symbol, a dictionary-based algorithm abbreviates recurring RLE tuples. For image decompression, the process chain needs to be run in reverse order performing the inverse operations.

3.1 Colorspace transformation

For the initial separation of the luminance, or luma, and chrominance, or chroma, information of the image, the input bitmaps undergo a transformation from the original RGB colorspace to a YUV representation (ITU, 2007). Each pair of multispectral pixels $R_iG_iB_i, i \in \{1, 2\}$ is processed according to equation 1 where c denotes the intensity range center (e.g., $c = 32768$ for 16 bits per sample) and $>>$ is the arithmetic shift right operator. For graylevel bitmaps, the luma values Y are obtained by subtracting c from each pixel value.

$$\begin{aligned} Y_i &= ((77R_i + 150G_i + 29B_i + 128) >> 8) - c \\ R_{12} &= (R_1 + R_2) >> 1 \\ G_{12} &= (G_1 + G_2) >> 1 \\ B_{12} &= (B_1 + B_2) >> 1 \\ U &= (-43R_{12} - 84G_{12} + 128B_{12} + 128) >> 8 \\ V &= (127R_{12} - 106G_{12} - 21B_{12} + 128) >> 8 \end{aligned} \tag{1}$$

Each output component gets written to a separate image plane. The resolution of the chroma components U and V is reduced by two in both image directions using averaging horizontally and the nearest neighbor filter vertically to at least partially suppress aliasing artifacts. Hence, the U and V samples are shared by a 2 x 2 subgrid of luma values which sometimes is called 4:2:0 subsampling in video coding. In the C++ library implementation, for input data that are band-interleaved by pixel (BIP), image traversal uses pointer arithmetic to increment the current position inside the bitmap and avoid full address recalculations. Pairwise RGB tuple processing eliminates expensive branching to differentiate between even and odd columns and likely will improve processor register utilization. Using bit shifts instead of divisions by the respective powers of two generally is faster and safe on unsigned sample data types. Also, the original intensity range will be translated but not left except for the calculation of intermediates. The Y, U and V outputs are signed integers of the same size as the input samples.

Image decoding employs the reverse transformation from equation 2 which operates on pairs of luminance samples to gain speed. For chroma upsampling, the U and V values remain constant for each 2 x 2 subgrid of Y values performing nearest neighbor interpolation. The R, G and B reconstructions must be clamped to the original image intensity range since over- or underflows may happen due to compression artifacts from other codec stages. For graylevel bitmaps, only the center c needs to be added to each Y value and fenced in to recover the unsigned intensity.

$$\begin{aligned} R_i &= clamp(c + Y_i + ((361V) >> 8)) \\ G_i &= clamp(c + Y_i + ((-90U - 184V) >> 8)) \\ B_i &= clamp(c + Y_i + ((456U + V) >> 8)) \end{aligned} \tag{2}$$

3.2 Frequency domain transformation

The Y, U and V image planes from the previous stage are individually decomposed into their frequency components by running the 2D discrete cosine transform (DCT) on 8 x 8 pixel blocks. The 2D DCT itself builds on a fast integer approximation of the floating-point 8-point 1D Loeffler DCT with 11 multiplications and 29 additions which is theoretically optimal (Loeffler et al., 1989). Figure 1 shows this DCT and its inverse, the IDCT, as signal flow graphs to be executed from left to right.

In the approximation of the transform, its original rotator blocks c1, c3 and c6 in their optimized form with three multiplications are replaced by c0, c$[16 \arccos(0.75)/\pi] \approx$ c3.6809 and c$[16 \arccos(1/(2\sqrt{2}))/\pi] \approx$ c6.1596, i.e., we rotate a little bit less and further. The involved factors a therefore yield 1, 0.75 and 0.5 respectively which eliminates two of them adding two arithmetic right shifts instead. The dependent constants $(b - a)$ and $-(a+b)$ subsequently are altered into close fractional powers of two to reflect the change. For the tiny $(b - a)$ value of

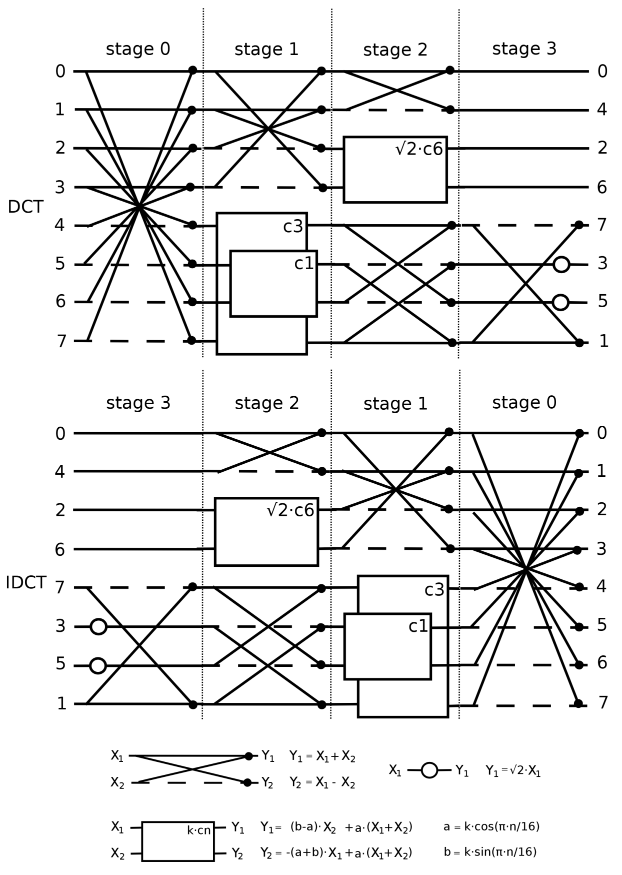

Figure 1. Signal flow graphs for the original 8-point Loeffler 1D DCT and 1D IDCT with 11 multiplications and 29 additions

function	a	(b-a)	-(a+b)
c1	0.9808	-0.7857	-1.1759
c3	0.8315	-0.2759	-1.3870
$\sqrt{2}$·c6	0.5412	0.7654	-1.8478
$\sqrt{2} \cdot x$	1.4142		
c0	1	-1	-1
c3.6809	0.75 (3/4)	-0.0886 (e) -0.0625 (-1/16) (f) -0.0938 (-3/32) (i)	-1.4114 (e) -1.4063 (-45/32)
$\sqrt{2}$·c6.1596	0.5 (1/2)	0.8229 (e) 0.8281 (53/64)	-1.8229 (e) -1.8281 (-117/64)
$\sqrt{2} \cdot x$	1.4063 (45/32) (f) 1.4375 (23/16) (i)		

Table 1. Original and integer frequency transform parameters to four decimals; for the latter, the exact (e) and power-of-two values for the forward (f) and inverse (i) operation are given

the c3.6809 approximation, the round-off error of its factor-free approximation of -1/16 asymmetrically gets compensated in the IDCT by -3/32. Similarly, the square root calculation of stage 3 will be modeled by the skew numbers of 45/32 for the forward transform and 23/16 for its inverse. As final coefficients that are directly applied to the output samples 3 and 5, these values do not appear in the DCT nor IDCT code but are incorporated as nominator-denominator pairs into the quantizer (see section 3.3.1). The quantizer will also take care of the scaling factor of 0.125 or 1/8 that comes with the forward Loeffler transform as depicted and which needs to be applied to preserve the amplitude of the samples.

In total, the adapted 8-point DCT and IDCT require 4 and 5 multiplications respectively plus 29 additions and 6 shifts for each direction. The reverse operation thus is slightly slower, however, this disadvantage will be overcompensated later by the lossless stages of the codec. Table 1 compares all factors of the floating-point 1D building block and their integer approximation as used by the library code.

To obtain the 2D DCT, the 1D DCT is run vertically on blocks of 8 x 8 samples of the input image planes whose dimensions must be aligned accordingly. The outcome is written to a small buffer of 64 intermediates on which the DCT gets executed horizontally. The final result is passed to the quantizer that writes the altered frequency coefficients into dedicated output image planes. For the 2D IDCT, the process is gone through in reverse order. Using a small temporary buffer eliminates unnecessary stride calculations to skip the remainder of image lines, keeps processor registers available to the DCT itself and is data cache friendly. Therefore, runtime will be optimized at the insignificant cost of a few hundred bytes of additional storage. The DCT code requires integer type promotion since the computed scaled coefficients very likely will exceed the signed input data type from color conversion except for trivial image content. Type transition happens once during the vertical pass.

Regarding the accuracy of the integer DCT, deviations to the original version will occur in both the forward and inverse direction due to the approximation and rounding. However, a quick test with roundtrips (DCT followed by IDCT) taken on 500 random 8 x 8 matrices made of signed 16 bit elements revealed that the mean absolute error caused by the frequency transform on average is around 0.3% of the number range. This value is considered acceptable for compression purposes.

3.3 Quantization

The obtained frequency components undergo quantization to reduce their amplitude to predefined discrete levels. This codec stage largely controls the degree of compression and hence is responsible for the bulk of image information loss to achieve practical data reduction ratios.

3.3.1 Forward and inverse quantization process Quantization itself involves integer divisions by positive numbers so that negligibly small DCT values become zero. For the inverse, or dequantization, the quotients are multiplied by the same numbers to roughly reconstruct the original frequencies. Depending on the rounding mode, the quantization error can be reduced trading accuracy for speed. Plain divisions that truncate towards zero are preconfigured in the library implementation. A slower but more precise rounding technique towards the closest integer which inevitably involves evaluating a condition is also available (figure 2).

```
// round n/d to closest integer the fast way
// for integer type T and d>0
template<typename T> T intDivRound(T n, T d) {

   // compiler will translate division by 2
   // into a shift and use a conditional move
   // to handle negative n in assembler code

   return (n+((n>=0) ? d : -d)/2)/d;
}
```

Figure 2. C++ snippet rounding DCT values to closest integer

For the 8 x 8 2D DCT blocks, quantization and dequantization are conducted using matrices of the same size. The matrices contain the divisors and factors respectively to be applied to the individual frequency components during compression and decompression. However, before being used, some of the raw matrix entries must be adjusted by the rational representations of $\sqrt{2}$ inherited from the DCT, and the DCT scaling of 1/8 needs to be incorporated. For the 2D DCT approximation, the original quantization matrices $\mathbf{Q_i}$ therefore have to be multiplied and divided element-wise by the 8 x 8 matrix products as shown by equation 3. This yields premultiplied ready-to-use versions $\mathbf{Q_i'}$ for the forward step and $\mathbf{D_i'}$ for dequantization.

$$\begin{aligned}
\mathbf{q_{mul}} &= \begin{bmatrix}1 & 1 & 1 & 45 & 1 & 45 & 1 & 1\end{bmatrix} \\
\mathbf{q_{div}} &= \begin{bmatrix}8 & 8 & 8 & 8{\cdot}32 & 8 & 8{\cdot}32 & 8 & 8\end{bmatrix} \\
\mathbf{Q_{mul}} &= q_{mul}^T \cdot q_{mul} \\
\mathbf{Q_{div}} &= q_{div}^T \cdot q_{div} \\
\\
\mathbf{d_{mul}} &= \begin{bmatrix}1 & 1 & 1 & 23 & 1 & 23 & 1 & 1\end{bmatrix} \\
\mathbf{d_{div}} &= \begin{bmatrix}1 & 1 & 1 & 16 & 1 & 16 & 1 & 1\end{bmatrix} \\
\mathbf{D_{mul}} &= d_{mul}^T \cdot d_{mul} \\
\mathbf{D_{div}} &= d_{div}^T \cdot d_{div} \\
\\
\mathbf{Q_i'} &= \mathbf{Q_i} \circ \mathbf{Q_{div}}/\mathbf{Q_{mul}} \\
\mathbf{D_i'} &= \mathbf{Q_i} \circ \mathbf{D_{mul}}/\mathbf{D_{div}}
\end{aligned} \tag{3}$$

Integrating the root and scale terms into the matrices $\mathbf{Q_i'}$ and $\mathbf{D_i'}$ limits the maximum dynamic range allowed for the image codec to $\log_2[2^{63}/(8^2 \cdot (8 \cdot 32)^2)] = 41$ bits per sample with 64 bit wide signed integers. This calculation is based on the maximum 1D DCT coefficient blow-up of eight, its scale of eight and the square root denominator of 32 to be applied as a factor to the quantization matrix. The premultiplied $\mathbf{Q_i'}$ and $\mathbf{D_i'}$ also impose a lower bound on the radiometric resolution of imagery to be compressed and decompressed. This issue is caused by significant round-off errors on small raw matrix entries and a coarse discretization on large raw matrix elements to be further scaled. Hence, in practice, the codec is not optimized for bitmaps with less than $\sim$12 bits per sample. This limitation however could be alleviated when the square root term is migrated back into the DCT at the cost of two more multiplications.

3.3.2 Base quantization matrices for bitrate control To be able to directly match the communication channel capacity, the missing quantization matrices $\mathbf{Q_i}$ are to be generated in such a way that the compression level can be expressed as a bitrate, i.e., in bits per image pixel (bpp). This makes them dependent on the entire compression algorithm, and hence no templates to be reused exist. The $\mathbf{Q_i}$ are therefore obtained through genetic optimization (Mitchell, 1998) on a set of representative training images. Optimization is performed for each bit depth to be supported by the codec expressed as bits per sample, or bits per component (bpc), which can be easily upscaled to the bpp value by the color channel count. Also, quantization matrices are refined separately for the luma and chroma image planes yielding matrix pairs $(\mathbf{Q_{i,l}}, \mathbf{Q_{i,c}})$. This reflects the greater influence of the brightness channel on visual image quality.

Like in biology, genetic optimization takes an initial population generated from random DNA strands and evaluates their adaptation to the environment with a fitness function. The fittest individuals reproduce and give birth to a new generation of population members by exchanging parts of their DNA while "weak" individuals get discarded. Diversity among the population members is further increased with point mutations, i.e., local random changes to their DNA. The evolution cycle repeats until a maximum generation count has been reached. The fittest individual of the population is eventually chosen to produce the final quantization matrix pair.

For the HDR image codec, the DNA independently defining the shape of the luma and chroma parts of the quantization matrix pairs is chosen as the arguments a and b of equation 4. Discrete sampling produces the $2 \cdot 64$ elements of $(\mathbf{Q_{i,l}}, \mathbf{Q_{i,c}})$. The created phenotypes loosely resemble the quantization matrix structure from JPEG for arbitrary bit depths and expose smooth transitions between the matrix entries.

$$z = a\sqrt{x} + a\sqrt{y} + b \qquad x, y \in \{0, ..., 7\} \tag{4}$$

The fitness function f to determine how well a quantization matrix pair suits a particular compression task is modeled as the square root of four arguments with empirically determined exponential weights (equation 5). Among the arguments is the ratio q_b of the currently achieved bitrate b_a to the target bitrate b_t which falls back to zero if the latter is exceeded by the former. Also, the universal image quality (UIQ) measure (Wang and Bovik, 2002) scaled to the $[0, 1]$ interval, peak signal-to-noise ratio (PSNR) (Salomon, 2006) and unique color count (UCC) are incorporated into f. The rationale for this choice is to obtain the best possible image quality (in terms of UIQ and PSNR) while approaching the given compression ratio (in terms of the bitrate quotient) as closely as possible from below. Also, the variety of colors (in terms of the UCC) shall be maximized and not get sacrificed to an excessively dominant luma channel.

$$\begin{aligned}
q_b &= \begin{cases} \frac{b_a}{b_t}, \; b_a \leq b_t \\ 0, \; else \end{cases} \\
f &= \sqrt{q_b^2 \cdot UIQ \cdot PSNR^2 \cdot UCC^{0.5}}
\end{aligned} \tag{5}$$

To evaluate the fitness function, the set of training images takes compression and decompression roundtrips for each quantization matrix pair of the population. The decompression outputs are compared to the respective originals in graylevel or RGB space. For each bitmap pair, PSNR is computed on a per-sample basis, and UIQ is obtained for each color band to be subsequently averaged. UCC calculation involves ordering the pixels of the decompression result by color in-place with the quicksort algorithm and counting the color transitions. This universal approach works for any sample type, band count and bit depth in linearithmic time.

The final fitness for a quantizer configuration $(\mathbf{Q_{i,l}}, \mathbf{Q_{i,c}})$ for RGB images and a particular target bitrate to be met is derived from the average of the fitness function arguments over all training data. When the genetic optimization is run for a sufficiently dense set of b_t, a lookup table from the achieved bitrates b_a (which ultimately will approach b_t) to the corresponding quantization matrix pairs can be constructed. For graylevel bitmaps, the optimization results from RGB imagery get recycled, and mappings from the partial bitrates of the luma channel to the luma quantization matrices $\mathbf{Q_{i,l}}$ are stored. Encoding images of the bit depth used during optimization to an almost arbitrary target bitrate is achieved by linear interpolation or extrapolation based on the two pairs of matrices that belong to the two neighboring bitrates according to the lookup table. Equation 6 displays sample raw luma and chroma matrices ($\mathbf{Q_{i,l}}, \mathbf{Q_{i,c}}$) for 16 bpc imagery for the target bitrate $b_t = 2$ bpp which equals a compression ratio of 1:24 for RGB data. The actually achieved Y, U and V partial bitrates from the genetic optimizer for this target are $b_a = 1.8615 + 0.0753 + 0.0626 = 1.9994$ bpp. This corresponds to image compression ratios of 24.0072 (RGB) and 8.5952 (graylevel).

$$\mathbf{Q_{i,l}} = \begin{bmatrix} 196 & 230 & 244 & 255 & 264 & 272 & 279 & 286 \\ 230 & 264 & 278 & 289 & 298 & 306 & 313 & 320 \\ 244 & 278 & 292 & 303 & 312 & 320 & 327 & 334 \\ 255 & 289 & 303 & 314 & 323 & 331 & 338 & 345 \\ 264 & 298 & 312 & 323 & 332 & 340 & 347 & 354 \\ 272 & 306 & 320 & 331 & 340 & 348 & 355 & 362 \\ 279 & 313 & 327 & 338 & 347 & 355 & 363 & 369 \\ 286 & 320 & 334 & 345 & 354 & 362 & 369 & 376 \end{bmatrix}$$
$$\mathbf{Q_{i,c}} = \begin{bmatrix} 420 & 424 & 425 & 427 & 428 & 428 & 429 & 430 \\ 424 & 428 & 429 & 430 & 431 & 432 & 433 & 433 \\ 425 & 429 & 431 & 432 & 433 & 433 & 434 & 435 \\ 427 & 430 & 432 & 433 & 434 & 435 & 435 & 436 \\ 428 & 431 & 433 & 434 & 435 & 436 & 436 & 437 \\ 428 & 432 & 433 & 435 & 436 & 436 & 437 & 438 \\ 429 & 433 & 434 & 435 & 436 & 437 & 438 & 439 \\ 430 & 433 & 435 & 436 & 437 & 438 & 439 & 439 \end{bmatrix} \quad (6)$$

3.3.3 Zone-adaptive quantization While the quantization matrices from genetic optimization will provide a reasonable starting point when a certain target bitrate needs to be met, the actually achieved compression ratios for individual images still may vary. Therefore, to closely approach the target, the matrices used to reduce the amplitude of the DCT components may get dynamically adjusted during the reduction process.

For this zone-adaptive quantization mechanism, equally sized strips of image scanlines are compressed individually. The bitrate actually achieved for the first strip is tracked, scaled to the full image dimensions and compared against the target bitrate. If the goal is exceeded, the bitrates to be used for the next image strips get gradually reduced based on the previous setting within a predefined corridor. On a shortfall, they will be increased recursively. The default adjustment factors are 0.8 and 1.24 for the bitrate decrease and increase up to the minimum and maximum of 0.5 and 4 respectively. In extreme cases, the compression ratio therefore may locally breathe between one quarter to two times the reduction rate aimed at. Due to the asymmetric adjustment factors (the reciprocal of 0.8 equals 1.25), the achieved average bitrate for the entire bitmap will have the tendency to stay slightly below the target bitrate. This behavior is desirable for data transmission with upper bandwidth limits.

3.4 Run-length encoding

Depending on the matrices used, the set of DCT coefficients likely will contain sequences of identical values after quantization. In fact, for realistic compression settings, the majority of frequency components form extended runs of zeros (see figure 3) which can be represented compactly through run-length encoding (RLE) without any information loss.

The run-length encoder of the C++ compression library by default operates on the concatenated stream of quantized DCT samples of all image planes. In contrast to JPEG where RLE is implemented on individual 2D DCT blocks with a zigzag-style reordering scheme, this approach potentially allows larger

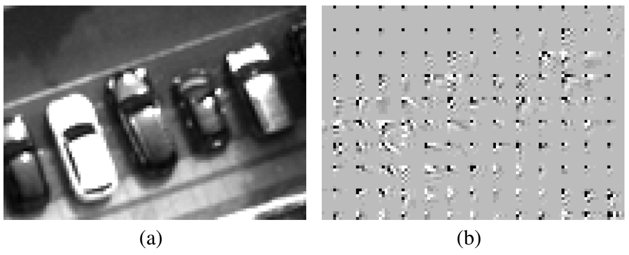
(a) (b)

Figure 3. Sample RLE input (a) luma channel of car scene, (b) quantized DCT coefficients, homogeneous gray indicates zeros

sequences of zeros to be reduced. Piecewise image decoding nevertheless remains possible using zone-adaptive quantization. The produced RLE output consists of (skip;value) tuples which consume just a few bytes each. The first part stores how many consecutive zeros precede a specific non-zero value. The value part can either be ”short” or ”long” to conserve memory on small quantized DCT coefficients. For images with 16 bits per sample, this translates into signed byte and signed short integer data types constituting a type demotion. Under- or overflowing frequency components will be truncated in the rare case the ”long” data type is insufficient. Discrimination between the two value sizes happens via the most significant bit of the skip part of the tuple. For 16 bit frames, the skip type is an unsigned byte, and hence up to 127 zeros can be fold at once. In the worst case, without any repeating DCT coefficients in the input stream like on near-lossless quantizer configurations, the RLE output will expand and not contract. This scenario currently is not specifically addressed by the HDR image codec.

3.5 Dictionary-based compression

The remaining redundancies in the tuple stream from the run-length encoder are further condensed with a second lossless compressor. Because of its speed, the LZ4 algorithm (Collet, 2016) (Collet, 2019) was chosen which is also integrated into the Linux kernel (Schmidt, 2017). LZ4 is a variant of the LZ77 dictionary-based encoders (Ziv and Lempel, 1977) that replace repeating sequences of (not necessarily identical) symbols with a single reference to their previous occurrence. To find the matches, an implicit dictionary formed by a sliding window over the most recent data gets examined.

For the proposed image codec, the high compression mode functions of the LZ4 open source reference library are invoked with a fixed reduction level of two. This setting offered a reasonable compromise between runtime and data reduction in combination with the other codec stages during a quick test. Nevertheless, while LZ4 in practice performs much faster than arithmetic or Huffman coding as used for JPEG, its compression ratio generally will stay behind these entropy-based algorithms.

4. PERFORMANCE ANALYSIS

As a proof-of-concept, the HDR transform coder was implemented as a C++ library whose functions are called by an accompanying demo application and the tool for the genetic optimization of the quantization matrices. The library does not utilize machine-specific instructions and hence can be compiled on different target platforms. Its code further is single-threaded. Parallel execution for the designated application of real-time mapping will be accomplished by running multiple codec instances concurrently on the images captured. To save memory for the intermediate bitmaps created by the HDR codec stages, each supported bit depth is associated a set of specific data types aggregated in a trait class. The algorithms involved in image encoding and decoding get instantiated with the respective trait at translation time. This enables type-specific compiler optimizations and boosts speed at the price of object code duplication.

To evaluate the achievable image quality and runtime for both compression and decompression, the library initially got configured with luma and chroma quantization matrices from genetic optimization. The dedicated optimizer tool was run on a diverse set of 16 RGB pinhole and mapped ortho images of 16 and 32 bits per component yielding 48 and 96 bits per pixel respectively. Except for computer-generated renderings, the data was originally captured by 14 bit sensors whose dynamic range was subsequently scaled with a non-linear transfer function to emulate true HDR hardware. Multithreaded calculation of the quantization matrices for the test bitmaps was conducted with a population size of 512 individuals and 128 generations. It took around three days on a fully utilized high-performance server with 56 physical processor cores and simultaneous multithreading (SMT) enabled even though downsampling to between 11 and 16 megapixels was applied to the input images in advance.

Optimizer data got recalled for performance analysis in full resolution ranging from 11 to 67 megapixels. Compression and decompression were carried out on the control and image processing unit of the DLR MACS-nano aerial camera (Kraft et al., 2023). MACS-nano fits into the nose section of a Quantum Systems Vector fixed-wing UAS and has been recently involved in real-time mapping missions on disaster sites (figure 4). The integrated computer builds on entry-level PC hardware with an Intel i3-1115G4E embedded processor that nominally operates at 3.0 GHz and 8 GiB of automotive LPDDR4 memory clocked at 1833 MHz. It runs the Linux operating system.

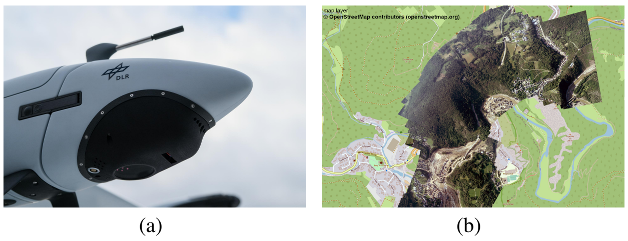

Figure 4. (a) MACS-nano in UAS nose, (b) Real-time mapping for damage assessment (Ahr valley flash flood, Germany, 2021)

Table 2 lists selected timings and bitrates plus the PSNR, UIQ and UCC quality readings as introduced in section 3.3.2 for a subset of four images of natural, urban and synthetic environments (figure 5). Measurements were also taken for the JPEG and JPEG 2000 codecs using release versions of the IJG libjpeg 9e (IJG, 2022) and OpenJPEG 2.5.0 (OpenJPEG Development Team, 2022) software libraries. Libjpeg was specifically compiled to handle raster data with 12 bits per sample. The dynamic range of the 16 bpc input imagery had to be linearly reduced after loading, and the fastest DCT option with Huffman entropy coding got chosen to ensure a fair comparison. Native bitmaps with 16 bits per component could be tested only for JPEG 2000 since the used accompanying command-line tools refused to read in 32 bpc bitmaps. The OpenJPEG utilities were invoked with the default compression and decompression settings.

For the 16 bpc bitmaps, the compression throughput of the proposed HDR image codec excluding I/O grows with the reduction rate. It increases from roughly 140 to nearly 400 megasamples per second (MS/s), i.e., 262 MiB/s to 750 MiB/s, mostly

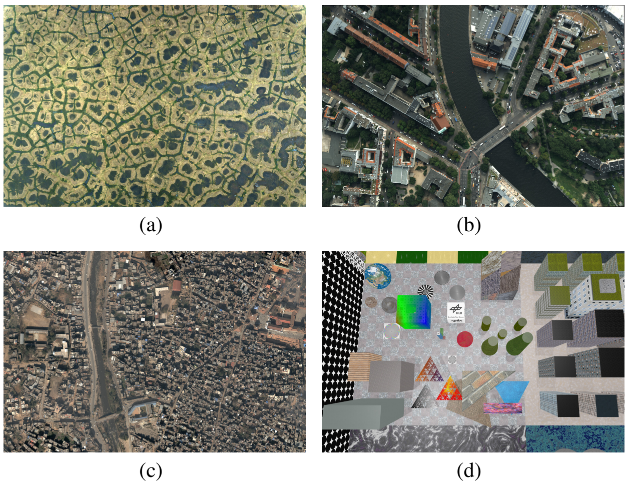

Figure 5. Subset of test imagery (cropped for display) (a) Drew Point/Alaska, (b) Berlin, (c) Kathmandu, (d) Synthetic Scene

because less data has to be handled by LZ4. The asymmetric runtime characteristics particularly of the lossless stages make decompression even faster. Decoding throughput always exceeds 300 MS/s and peaks at 486 MS/s or nearly 1 GiB/s of data. In both disciplines, the reference JPEG implementation is outperformed by almost a factor of two. The popular JPEG 2000 library at best is roughly 60 times slower during compression and five times behind on decompression. For 32 bpc imagery, less megasamples are processed per second in absolute numbers. However, the larger sample type increases the byte count that can be compacted per unit of time by 46% to 75% for comparable bitrates. At reduction settings of 1:24 and 1:48, frame rates greater than 1.5 Hz and 1.8 Hz are reached respectively by the HDR encoder for the 50 megapixel bitmaps of both color depths. The resulting data streams can be transmitted over 4G networks considering their nominal data rates (Dahlman et al., 2013). In the special case of on-board-mapping, when overlapping image parts get clipped during the projection onto a surface model to remove redundancies in advance (Hein and Berger, 2018), the achievable frame rates will further increase. A gain linear to the pixel count reduction could be expected.

Regarding image quality, the HDR codec introduces more artifacts than the established compression standards on the 16 bpc bitmaps. PSNRs for the urban Berlin scene are lower by about 5 dB compared to the 12 bit-enabled JPEG library for quality factors of $q = 20$ and $q = 4$ to achieve equal bitrates of $b_t = 2$ and $b_t = 1$ bpp. The delta to wavelet-based JPEG 2000 is roughly 8 dB. The unique color counts of the proposed approach and JPEG are almost on par. Despite the drop to approximately 39% in the worst case for the subset, the UCC remains well into the millions. Eight-bit tone mapping solutions that allow standard JPEG streams to be transmitted likely will reproduce much less color shades, e.g., 575046 for the Berlin photo when applying the GIMP open-source image processor. JPEG 2000 manages it to preserve and slightly extend the original chromaticity. UCC readings beyond 100% of the original value also appear with the HDR codec on high bitrates. These overshots can be attributed to inaccuracies in the frequency transforms and quantization errors. For all compressors, the UIQ values approach the theoretical maximum and reflect the PSNR behavior. Figures 6 and 7 visualize the achieved image quality. Fine contrasts are preserved by all codecs for medium bitrates. With stronger compression, block artifacts characteristic for DCT-based methods become noticeable especially for the HDR image encoder.

image	reduction		compress time			decompress time			PSNR	UIQ	UCC		
	bpp	**ratio**	**ms**	**MS/s**	**MiB/s**	**ms**	**MS/s**	**MiB/s**	**dB**	**[0,1]**	**orig #**	**rndtrip #**	**%**
Proposed HDR codec, 16 bpc RGB (48 bpp)													
Drew Point	6.04	7.95	335	140.78	268.52	148	318.66	607.79	34.47	0.9915		15707328	105.28
4864x3232	2.22	21.61	178	264.95	505.36	116	406.56	775.46	28.94	0.9690	14920049	14190133	95.11
16 Mpix, 90 MiB	1.00	48.08	121	389.76	743.41	98	481.24	917.89	26.54	0.9455		9847884	66.00
Berlin	5.25	9.14	309	152.63	291.11	140	336.87	642.52	39.45	0.9983		9944454	64.02
4864x3232	2.12	22.69	170	277.42	529.14	108	436.68	832.90	35.87	0.9960	15533264	10974934	70.65
16 Mpix, 90 MiB	1.00	48.01	120	393.01	749.61	97	486.20	927.35	32.21	0.9907		6775190	43.62
Kathmandu	5.88	8.16	1465	137.42	262.12	620	324.72	619.36	37.54	0.9966		60253954	91.99
8192x8192	2.11	22.70	738	272.80	520.33	462	435.77	831.17	34.40	0.9937	65503284	49454824	75.50
67 Mpix, 384 MiB	1.00	47.85	537	374.91	715.08	421	478.21	912.11	32.25	0.9901		36732520	56.08
Synthetic Scene	5.41	8.88	973	155.19	296.00	419	360.37	687.35	35.56	0.9976		25029684	49.76
8192x6144	2.05	23.38	548	275.54	525.55	354	426.54	813.56	31.57	0.9938	50304958	29393152	58.43
50 Mpix, 288 MiB	1.03	46.47	416	362.97	692.31	326	463.18	883.44	28.73	0.9877		19701873	39.16
Proposed HDR codec, 32 bpc RGB (96 bpp)													
Drew Point	6.99	13.73	410	115.03	438.80	165	285.83	1090.34	33.95	0.9904		15703298	105.25
4864x3232	2.36	40.65	222	212.44	810.39	119	396.31	1511.82	28.78	0.9679	14920049	13533093	90.70
16 Mpix, 180 MiB	1.12	86.00	165	285.83	1090.34	103	457.88	1746.66	26.37	0.9432		8852251	59.33
Berlin	6.28	15.27	353	133.60	509.65	138	341.75	1303.67	39.79	0.9984		14405587	92.74
4864x3232	2.00	47.90	199	237.00	904.05	109	432.67	1650.52	35.34	0.9956	15533264	9994728	64.34
16 Mpix, 180 MiB	1.05	91.70	163	289.33	1103.72	104	453.47	1729.87	32.41	0.9913		5988145	38.55
Kathmandu	6.76	14.20	1678	119.98	457.69	609	330.59	1261.08	37.92	0.9971		62749313	95.80
8192x8192	1.98	48.58	877	229.56	875.71	475	423.85	1616.84	33.79	0.9930	65503284	45423919	69.35
67 Mpix, 768 MiB	1.08	88.77	728	276.55	1054.95	441	456.52	1741.50	31.40	0.9880		31336736	47.84
Synthetic Scene	6.28	15.29	1150	131.30	500.87	457	330.41	1260.39	35.51	0.9975		44767833	88.99
8192x6144	1.98	48.55	649	232.66	887.52	360	419.43	1600.00	30.86	0.9925	50304958	29219272	58.08
50 Mpix, 576 MiB	1.06	90.89	529	285.44	1088.85	333	453.44	1729.73	28.51	0.9869		18601917	36.98
IJG libjpeg v9e (JPEG), 16 bpc ↓ 12 bpc RGB (48 ↓ 36 bpp)													
Berlin (q=20)	2.08	23.08	349	135.13	257.75	199	236.99	452.03	40.64	0.9987	15533264	10833048	69.74
Berlin (q=4)	0.96	49.87	284	166.06	316.74	172	274.19	522.98	37.20	0.9970		5632683	36.26
OpenJPEG (JPEG 2000), 16 bpc RGB (48 bpp)													
Berlin (cr=24)	2.00	24.01	10042	4.70	8.96	814	57.94	110.51	43.75	0.9993	15533264	15670555	100.88
Berlin (cr=48)	1.00	48.01	9980	4.73	9.01	540	87.34	166.58	39.50	0.9983		15493831	99.75

Table 2. HDR image codec results on the image subset for target bitrates of b_t = 6, 2 and 1 bits per pixel (bpp) in comparison to popular JPEG and JPEG 2000 implementations

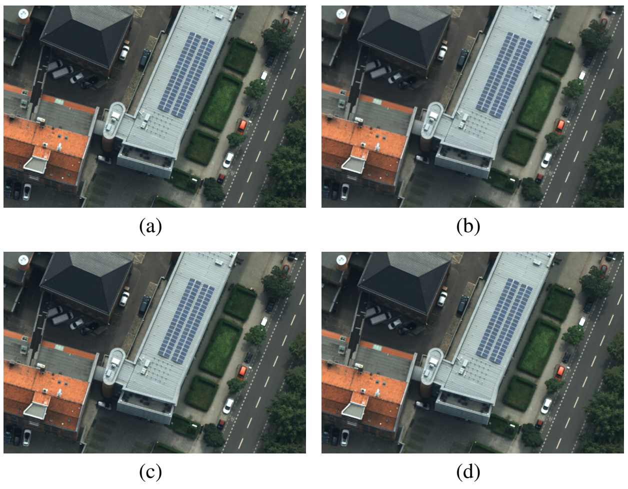

Figure 6. Image quality comparison for the Berlin scene, $b_t = 2$ bpp (1:24) (a) original, (b) HDR codec, (c) JPEG, (d) JPEG 2000

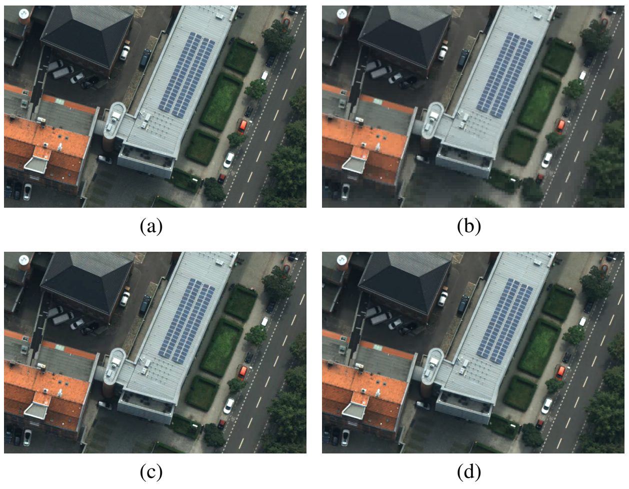

Figure 7. Image quality comparison for the Berlin scene, $b_t = 1$ bpp (1:48) (a) original, (b) HDR codec, (c) JPEG, (d) JPEG 2000

Quality and compression ratio control based on the mapping from bitrates to quantization matrices from genetic optimization in combination with zone-adaptive image encoding seems to work as expected. The actually obtained bitrates mostly stay close to the targets of $b_t = 6$, 2 and 1 bpp. Deviations concentrate on low reduction ratios for which fewer quantization matrix samples have been generated and stored inside the lookup table. Although the perfection of JPEG 2000 remains unmatched, the accuracy obtained with the HDR image codec will enable on-line adjustments of the compression degree to the capacity of the actual communication downlink independently of the upcoming image content to be transmitted.

5. CONCLUSION

This paper has outlined the design and implementation of a fast transform coder for HDR real-time mapping applications that runs on affordable off-the-shelf general purpose computing hardware. The proposed codec covers a wide range of bit depths to which it can be specifically adapted using genetic optimization. The portable proof-of-concept library that encapsulates the underlying highly optimized subalgorithms is capable of compressing and decompressing 50 megapixel RGB bitmaps of 16 and even 32 bits per component in less than a second on a single CPU core. Runtime outperforms optimized reference implementations of the JPEG and JPEG 2000 standards at the cost of an acceptable level of image quality degradation. Depending on the image size and intended frame rate, the achieved compression ratios will enable real-time transmission of the compressed data streams over decent radio downlinks and cellular networks.

REFERENCES

Belyaev, E., Mantel, C., Forchhammer, S., 2017. High bit depth infrared image compression via low bit depth codecs. *Infrared Remote Sensing and Instrumentation XXV*, 10403, SPIE International Society for Optics and Photonics.

CCSDS, 2017. Image Data Compression - Recommended Standard CCSDS 122.0-B-2. Consultative Committee for Space Data Systems (CCSDS), CCSDS Secretariat, NASA, Washington, D.C., USA.

Collet, Y., 2016. LZ4 Frame Format Description. github.com/lz4/lz4/blob/dev/doc/lz4_Frame_format.md (23 April 2023).

Collet, Y., 2019. LZ4 Block Format Description. github.com/lz4/lz4/blob/dev/doc/lz4_Block_format.md (23 April 2023).

Dahlman, E., Parkvall, S., Skold, J., 2013. *4G: LTE/LTE-Advanced for Mobile Broadband*. 2nd edn, Academic Press, Cambridge, MA, USA.

Hein, D., Berger, R., 2018. Terrain Aware Image Clipping for Real-Time Aerial Mapping. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-1, 61–68.

IJG, 2022. Libjpeg 9e. Independent JPEG Group. ijg.org (27 April 2023).

ISO, 2004. Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification. Standard, International Organization for Standardization, Geneva, Switzerland.

ISO, 2019. Information technology – JPEG 2000 image coding system – Part 1: Core coding system. Standard, International Organization for Standardization, Geneva, Switzerland.

ISO, 2020a. Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 2: High efficiency video coding. Standard, International Organization for Standardization, Geneva, Switzerland.

ISO, 2020b. Information technology – Scalable compression and coding of continuous-tone still images – Part 1: Core coding system specification. Standard, International Organization for Standardization, Geneva, Switzerland.

ITU, 2007. BT.601: Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios. Standard, International Telecommunication Union, Geneva, Switzerland.

Kraft, T., Meißner, H., Geßner, M., Gäde, J., Brauchle, J., Hein, D., Gonschorek, J., Helmrich, J., Bayer, S., Berger, R., 2023. Hybride UAV-Systeme: Technologie und Anwendungen. DVW e.V. (ed.), *UAV 2023 - Geodaten nach Maß*, DVW-Schriftenreihe 105, Wißner-Verlag, Augsburg, Germany, 61–73.

Loeffler, C., Ligtenberg, A., Moschytz, G. S., 1989. Practical fast 1-D DCT algorithms with 11 multiplications. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2, 988–991.

Mantel, C., Forchhammer, S., 2017. Compression of Infrared Images. *Electronic Imaging*, 2017, 21-26.

Manthey, K., 2014. A New Real-Time Architecture for Image Compression onboard Satellites based on CCSDS Image Data Compression. *On-Board Payload Data Compression Workshop*, Venice, Italy.

Melián, J. M., Jiménez, A., Díaz, M., Morales, A., Horstrand, P., Guerra, R., López, S., López, J. F., 2021. Real-Time Hyperspectral Data Transmission for UAV-Based Acquisition Platforms. *Remote Sensing*, 13(5).

Mitchell, M., 1998. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA.

OpenJPEG Development Team, 2022. OpenJPEG 2.5.0. Image and Signal Processing Group, Université de Louvain. openjpeg.org (27 April 2023).

Pennebaker, W. B., Mitchell, J. L., 1992. *JPEG Still Image Data Compression Standard*. Kluwer Academic Publishers, Norwell, MA, USA.

Salomon, D., 2006. *Data Compression: The Complete Reference*. Springer-Verlag, Berlin/Heidelberg, Germany.

Schmidt, S., 2017. Implementierung von LZ4Fast im Linux-Kernel. Technical report, Universität Hamburg.

Wang, Z., Bovik, A., 2002. A universal image quality index. *IEEE Signal Processing Letters*, 9(3), 81-84.

Ziv, J., Lempel, A., 1977. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3), 337-343.