JPEG 2000: Guide for Digital Libraries

David Barina*   Ondrej Klima

June 5, 2020

Brno University of Technology
Faculty of Information Technology
Centre of Excellence IT4Innovations
Bozetechova 1/2, 612 66 Brno, Czech Republic

Abstract

The JPEG 2000 image compression standard is being used in many areas, such as for the cultural heritage preservation. Unlike the preceding JPEG method, JPEG 2000 compression is governed by a large number of parameters that control its effectiveness. Some of them are constrained by specific needs of particular use cases, others allow controlling (1) resource demands and (2) rate–distortion trade-off. The involved users usually choose these parameters according to some guidelines or analyses. However, inappropriate adjustment of parameters may easily lead to compression that is optimal neither from the perspective of (1) nor (2). This paper serves as a guide for the preservation of digital heritage in cultural heritage institutions, including libraries, archives, and museums.

Keywords: Cultural heritage preservation, Lossy compression, JPEG 2000

1 Introduction

Since most fields in engineering deal with enormous amounts of digital images, lossy image compression became one of the most popular problems in computer science. In this context, one must trade off two competing costs: the number of bits of the compressed image (rate) and the error arising from the loss of information (distortion). Different applications impose different rate–distortion trade-offs and also additional constraints on computing demands, scalability, error resilience, etc. The JPEG 2000 represents both preservation and access format addressing all these impositions in a single internationally-standardized framework. Especially, it is of growing interest to the cultural heritage archive community as a better alternative to TIFF format [1].

Although this article is devoted to the JPEG 2000 format, we start with the description of its older predecessor – the JPEG standard [2]. Apart from use-specific restrictions like colour transform and subsampling, the original JPEG method is parametrized by 64-element quantization table, which must be specified by the application (or user) as an input to the encoder. The purpose of the quantization matrix is to specify the trade-off on the rate–distortion curve. Each element specifies the step size of the quantizer for its corresponding DCT coefficient. Usually, the end user is not mandated to specify the 64-element table directly. Instead, he is being asked to enter a dimensionless number indicating the rate–distortion trade-off, target bit-rate, or some quality metric. Creating the quantization table is a matter of the particular codec used. To be fair, the JPEG also provides some additional options like using restart markers, progressive transmission mode, and possibly others, depending on the codec.

Unlike JPEG, the JPEG 2000 compression is governed by a large number of parameters. These options especially affect rate–distortion trade-off, computational and memory demands, error resilience, accessibility of image parts, or method of progressive transmission. The chosen values for some of these parameters are restricted by a specific use case. This is typically the case of bit depth or the choice between lossy and

*Corresponding author.

lossless compression. The others must be specified by a user, according to his requirements. This may include choosing image tiling, a number of image resolutions, and inserting various markers to facilitate fast random access or error resilience. In this article, we discuss the merits of the choices for fundamental parameters and, wherever possible, provide either a recommendation or commonly used value.

The paper is specially designed for users who plan to manage large volumes of image data. These include, for example, museums, cultural archives, and libraries that carry out mass digitization. In the process of mass digitization, financial means and storage capacity are often considered crucial. This is the primary reason why the JPEG 2000 gains in popularity at these institutions. On the other side, it is also considered as a big risk for the future accessibility of the material.

2 JPEG 2000 Parameters

The following paragraphs review the most important parameters of the JPEG 2000 compression and provide guidance to choose values suitable for a particular application. Mostly, the chosen values also depend on the specific data. So no predefined settings (profiles) for all the parameters below exist. However, we provide a recommendation and mention the commonly used value for each of the discussed parameters. We also demonstrate some of the phenomena occurring in the compression using the test image in Figure 1. At the end of the section, the parameters are related to two most commonly used software codecs.

The JPEG 2000 compression is built upon multi-resolution image representation provided by the two-dimensional discrete wavelet transform (DWT). This article expects a certain knowledge of the terminology surrounding this image-processing tool. An extensive introduction to this area is given in [3]. The article also assumes a basic knowledge of image-compression terminology. In this case, we refer the reader to [4].

The technical review of the JPEG 2000 was already given many times elsewhere, e.g., in [4, 5]. In this paper, we focus on the end-user perspective. Unlike its competitors (mainly TIFF and JPEG), the key feature of the JPEG 2000 is to allow the end-user to specify multiple bit-rates, on which the image is accessible. As a direct consequence, the image can be decoded with a variety of distortions, ranging from the worst to the best quality, using a single compressed file. This aspect is illustrated in Figure 2. The file mostly refers to the JP2/JPX container. This container carries the JPEG 2000 code-stream, which eventually carries the actual image data. Besides such a fundamental hierarchy, other important aspects need to be understood by the end user to properly exploit the format features. Discussion on the parameters follows.

Comparing compression results requires a measure of image quality. The most commonly used measure is the peak signal-to-noise ratio (PSNR). Unlike the mean-squared error, the PSNR does not depend on pixel intensities. However, the PSNR is only meaningful for comparing compression results for the same image. Inter-image comparison of the PSNR is meaningless. The PSNR is usually expressed in terms of the logarithmic scale (in decibels).

Wavelet and compression path. In the beginning, recall that this paper focuses solely on lossy image compression. JPEG 2000 offers a choice of either the reversible and irreversible compression path. The reversible path employs the integer approximation of the CDF 5/3 wavelet to decompose the compressed image into multiple resolutions. On the contrary, the irreversible path uses the real CDF 9/7 wavelet for the same task. Both of the wavelets can be used for lossy compression. However, as also documented in [5], the CDF 9/7 consistently outperforms the 5/3 wavelet with the performance gap increasing with increasing bit-rate.

Considering digital libraries and other memory institutions, the reversible (lossless) path is suitable for the preservation of archival (master) copies, whereas the irreversible (lossy) path is suitable for production copies. Moreover, the irreversible path is especially suitable for connection with multiple quality layers (discussed below).

Colour transform and subsampling. JPEG 2000 allows for RGB to YCbCr colour transform. Since energy in RGB is more evenly distributed across the three components, some of the redundancy between the original RGB components can be exploited in this way. According to [4], using the YCbCr conversion, the

Figure 1: The test image used to demonstrate various dependencies in this paper. It is a map of the Margrave of Moravia and the Duchy of Silesia. Digitized at a resolution of 6486 × 5376 at 24 bits per pixel. Available in [6].

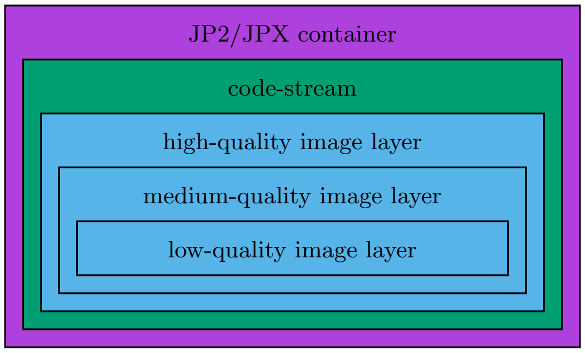

Figure 2: User view of the JPEG 2000 format hierarchy. Quality layers allow accessing the image at multiple bit-rates.

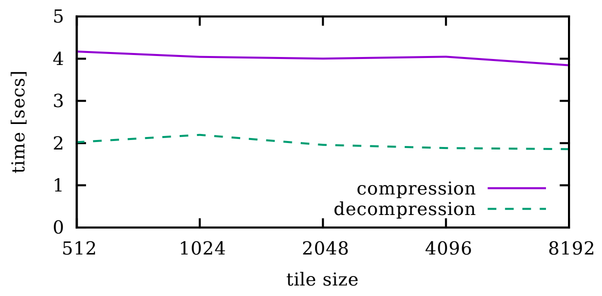

Figure 3: A relationship between the tile size and the corresponding compression/decompression time. Tiles are square in shape. The image was compressed at 0.8 bits/pel, using 5 decomposition levels and 64×64 code-blocks.

two chrominance components typically account for less than 20 % of the bits in the compressed image. For common images, compression in YCbCr exhibits a higher compression performance.

Together with colour transform and bit depth, the chroma subsampling is commonly determined by the use-specific restrictions. Here we would just point out that although even 4:2:2 subsampling is commonly referred to as visually lossless, it is not valid in all circumstances. Artefacts such as non-constant luminance can occur if chroma subsampling is used. Further explanation can be found in [7].

Because the human visual system is much more sensitive to the luminance than chrominance, the use of YCbCr conversion as well as chroma subsampling makes great sense for common use. It should also be noted that the vast majority of documents in digital libraries have the character of black text on a light background, which further enhances the sense of the YCbCr conversion.

Tile size. The first step in compression is to partition the input image into rectangular and nonoverlapping areas of equal size, so-called tiles. The tiles are then compressed independently, causing block artefacts on tile boundaries. The tile size is arbitrary and can contain the whole image. As demonstrated in [5], the compression performance decreases with decreasing tile size. Please note that, occasionally, one can face the false assertion claiming that the minimum tile size is 128×128 samples. This is by no means true, except that maximum number of allowable tiles is 65 535 for any JPEG 2000 code-stream.

The tiling is particularly useful for memory restricted environments. An efficient implementation of a JPEG 2000 codec is analyzed in [8]. The total memory demands of the codec can be expressed as $(3 \times 2^{c_n} + K)M$ samples, where 2^{c_n} is code-block height, K is a small constant, and M is the image width. This relationship allows comparing the employed tile size to the working set size, which should fit into a CPU cache. A nice introduction into CPU caches can be seen in [9]. It follows that current commonly used systems suit the use up to $4\,096 \times 4\,096$ tiles. Moreover, recommendations on the minimum tile size can be found in the literature. For example, the authors of [10] suggest the minimum size of 256×256 to avoid blocking artefacts.

An efficient implementation of the two-dimensional discrete wavelet transform takes only linear time. This means that the running time increases linearly with the size of the input, regardless of the tile size. So the assumption that a smaller tile size means a faster compression or decompression is not usually valid. In detail, the efficiency of individual JPEG 2000 software codecs has been studied in [11]. The authors found that the most widely used reference codec, OpenJPEG, implements the DWT in a very suboptimal way. On the contrary, the most widely used proprietary codec, Kakadu, exhibits the expected linear time complexity.

To demonstrate the independence of codec's speed to a tile size, we compressed a large enough image with different tile sizes. Figure 3 shows an obtained relationship for both compression and decompression. Note the asymmetry in the speed of compression and decompression. All of the examined sizes offer virtually indistinguishable rate–distortion performance. Here and in other experiments in this paper, the Kakadu codec was used, as it provides optimal time complexity with respect to input size.[1] Unless stated otherwise,

[1]Kakadu 7.10 running on Linux in 12 threads on Intel Xeon E5-2620 v3.

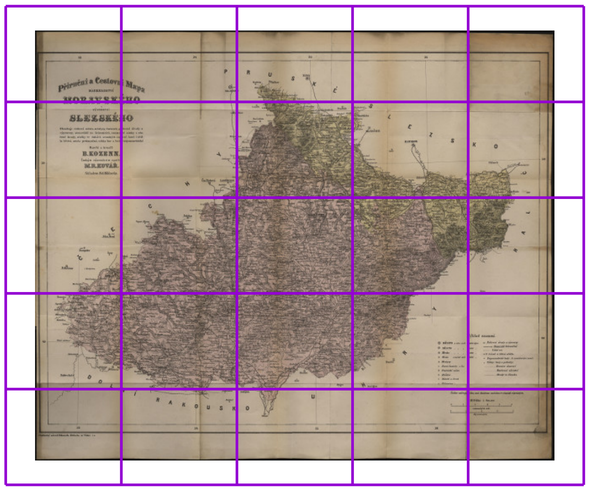

Figure 4: Sample image tiling. All tiles must be of the same size. The tile size can however be arbitrary and the tile grid and image origins do not need to be aligned. The tiles are then transformed and encoded independently.

we use code-blocks 64 × 64, tiles of the size 4 096 × 4 096, and 5 decomposition levels.

As the size of the CPU cache and memory increases, we safely recommend using the tile size of 4 096 × 4 096 pixels. Smaller sizes make little sense today, while larger sizes could cause performance bottleneck in bulk processing.

Image and tile origin. Besides the size of the tiles, JPEG 2000 allows specifying a tile and image origin, i.e. an offset to zero coordinates. To better understand these possibilities, see Figure 4. Observe especially the possibility to set non-zero image origin while keeping the tiles anchored at (0,0). The impact on compression performance is negligible. Along with the tile size, this feature can be exploited to various tricks, such as cropping, independent access to or settings for specific parts of the image. Aside from such intentions, the most commonly used value is (0,0) for both an image as well as tile origin.

Number of decomposition levels. The discrete wavelet transform is a type of pyramid image representation, in which the image is decomposed into multiple resolutions. JPEG 2000 encoders allow specifying either the number of the resolutions or decomposition levels. The number of resolutions is one more than the number of decomposition levels. Zero levels imply no transformation and a single resolution. Anyhow, the number of decomposition levels affects the compression efficiency as well as the number of resolutions at which the image can be decompressed. In practice, this number also affect the compression complexity. Many papers suggest five or six decomposition levels to be the best universal choice. Although it is commonly said that a higher number of levels means a better compression ratio, it is not completely true. Another recommendation that can be found in the literature, e.g., [12], suggests the use of as many levels as are needed to create a conveniently-sized thumbnail at the lowest resolution.

To illustrate a rate–distortion relationship and compression speed for different decompression levels, the test image was compressed at three common bit-rates. Figure 5 shows the results. It can be seen that further decomposition levels improve both the signal-to-noise ratio and the compression time to some extent. Thus the suggested five or six levels should be the right choice for our image. However, for the best results, the number of decomposition levels must be determined from specific data. Otherwise, the chosen number may easily lead to a suboptimal performance in terms of compression efficiency and complexity. Note also increasing speeds with an increasing compression ratio, which corresponds to the decreasing number of bits that need to be processed.

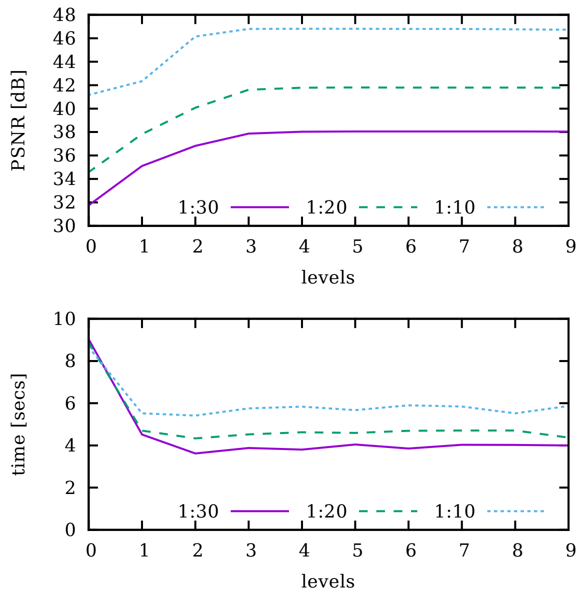

Figure 5: A dependency of a distortion (top) and compression speed (bottom) on decomposition levels.

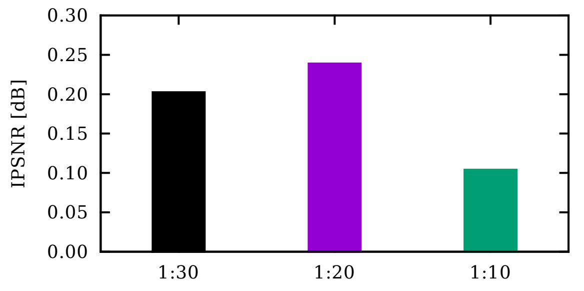

Figure 6: An improvement of the PSNR for the code-blocks 64 × 64 over code-blocks 32 × 32. Compressed at three different compression ratios.

We recommend always experimentally determine which number of decomposition levels gives the best results for your data. If this is not possible (e.g., data is not known in advance) use the recommended five decomposition levels.

Code-block size. In JPEG 2000 coder, the image is decomposed into a collection of wavelet coefficients, known as subbands. These subbands are then partitioned into rectangular blocks, known as code-blocks, for the purpose of independent coefficient coding. As a consequence, spatial accessibility in JPEG 2000 is based on the fact that each code-block is associated with a relatively small spatial region. Code-block size has to be a power of two and the total number of samples in a code-block cannot exceed 4 096. Commonly used code-block sizes are 32 × 32 or 64 × 64 samples. Note that both of these also conform code-stream restrictions given by Profile-0 and Profile-1 defined in the standard [13]. The influence on compression performance was investigated in [5]. As demonstrated in Figure 6, the larger code-blocks provide higher compression performance. Also, compression/decompression is faster for a larger code-block size. See the compression speed for our test image in Figure 7. Thus, if no other restrictions exist, use the largest possible code-block size, which is 64 × 64. The size of 32 × 32 makes no sense today.

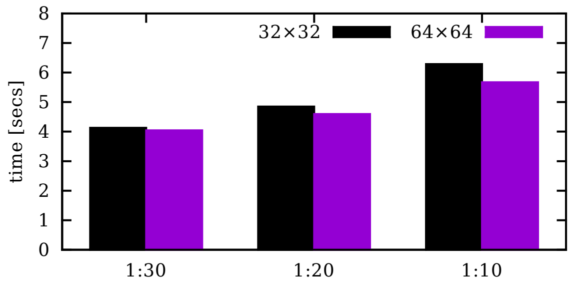

Figure 7: Compression time for code-blocks 32×32 and code-blocks 64×64.

Table 1: The relationship between the bit-rate (bits/pel) and compression ratio for color images.

bit-rate	ratio
12.0	2:1
2.4	10:1
1.2	20:1

Bit-rate. When compressing natural images, images compressed by JPEG 2000 are usually judged visually lossless, e.g., in [12], at a bit rate of around 1 bit per pixel. Below this rate, compression artefacts become more noticeable. Beware that some software libraries are instructed by the bit-rate, whereas others by compression ratio, compressed file size, or compression factor. See Table 1 for better understanding. On the other hand, the libraries usually also provide compression by quality, specifying either the PSNR, a slope on the rate–distortion curve, or quality factor.

In terms of the PSNR, good-quality decompressed images typically have values of 30 dB or more. In Kakadu encoder, the slope values can range from 0 (maximum quality) to 65 535 (all data discarded). As noted in [14], the useful operating range for this parameter is roughly between 49 000 and 53 000.

For production copies, we would recommend using a bitrate of about 1 to 2 bits per pixel (bpp) together with multiple logarithmically-spaced quality layers (discussed below). So smaller bitrates (0.5 bpp, 0.25 bpp, and so on) will also be available in the coded data stream.

Quality layers. Quality layers in JPEG 2000 allow specifying multiple bit-rates (compression ratios) for a single image and thus provide the possibility of progressive quality improvement or post-compression quality selection. In more detail, the final code-stream is composed of a succession of layers, where each subsequent layer achieves successively higher target bit-rates and supposedly contributes to a higher quality image. This scalability can be useful in networked environments, where each subsequently received quality layer contributes to the quality of the image which is displayed on the client side. If such scalability is irrelevant, only a single quality layer may be used. Otherwise, as discussed in [5], the layers should be more finely spaced at the lower bit-rates (i.e. spaced logarithmically). This recommendation is based on the fact that at lower bit-rates, a small increment in bit-rate has more impact on quality than the same incremental change at higher bit-rates. Naturally, in some applications, there may be a small known set of bit-rates at which the quality layers may be targeted. Using layers adds overhead that slightly increases the size of the compressed file. However, as found in [15, 16], this overhead is judged insignificant in comparison to their advantages. The typical number of quality layers used ranges from 5 to 25. If you are not sure how many layers to use here, you can use 12 layers of quality as the golden middle way.

Progression order. The JPEG 2000 allows changing the order (of the packets) in which the quality, resolution, spatial extent, and colour components increase. The order in which these packets appear in the code-stream is called the progression order. The progression order can be determined independently for each tile. All the packets for a tile can be ordered by using nested loops, which iterate over quality layers (L), resolutions (R), spatial position (P), and colour components (C). The JPEG 2000 allows five progression orders changing the nesting order of the above loops. These progression orders are identified as the LRCP (progressive by quality), RLCP (progressive by resolution, then by layer), RPCL (progressive by resolution, then by position), PCRL (progressive by position), and CPRL (by component). The LRCP order is commonly recommended, assuming that we want the best quality as a function of bit-rate. On the other hand, when the decompressing speed is our concern, layer-last progressions (RPCL, PCRL, and CPRL) used in conjunction with PLT/PLM marker segments are preferred. These progressions simplify the efficient management of code-block data in a decoder. Considering the layer-last progressions, the RPCL is preferred for interactive viewing of large images. The PCRL order may be favorable for compressing large untiled images. More extensive comments on the progressions orders can be found in [5].

The above-described progression order is defined at the level of a tile. Furthermore, the tiles can be broken into smaller pieces, referred to as the tile-parts, that are mutually interleaved. In that way, the concept of progression is extended to the entire image. The tile-parts can be introduced in such a way that each one consists of packets from only one resolution level (R), only one quality layer (L), or only one component (C).

The granularity with the spatial extent is gradually increased is controlled be the precinct size. For example, the precinct size chosen so that an entire subband belongs to a single precinct implies very poor spatial accessibility. On the other hand, the overhead due to the introduction of precincts is very small. This size may be different on different resolutions, and it is restricted to be a power of two. Note that each precinct boundary coincides with a code-block boundary.

Fast random access. The PPM, PLM, and TLM marker segments can be useful for fast random access into the code-stream. The TLM (tile-part lengths: main header) segments facilitate random access by describing lengths of every tile-part. Using this information, every tile-part of a given tile can be quickly located and extracted from the code-stream. Without this segment, a decoder would need to read a header for the first tile, parse its length, and seek to the second tile, etc.

The PLM (packet lengths: main header) marker segment is useful for random access into the code-stream at a finer granularity than that provided by TLM segments. These segments record the lengths of packets. With this information, a decoder can quickly determine the offset to each packet. The price of this finer granularity is somewhat higher overhead.

Finally, the PPM (packed packet headers: main header) segment can be used to collect and relocate all packet headers to the main header of the JPEG 2000 code-stream. This can come in handy for certain types of fast random access. Unfortunately, the last two marker segments are rarely supported by encoders.

Since the PLM and PPM segments are usually not supported, we recommend using at least the TLM marker segments so the image parts can be accessed randomly, whereas the overhead is small.

Error resilience. The JPEG 2000 defines error resilient tools on several levels. However, their exploitation fully depends on the decoder used.

On coarse packet level, the SOP and EPH markers together with small precincts are useful for the purpose of error resilient decoding. The SOP marker segments and EPH markers are used for resynchronization of the decoder, for the cost of low overhead. In addition to error resilience, they can be used to quickly locate packets and packet headers. The SOP (start of packet) marker segment can be included in front of some or all packets. Conversely, the EPH (end of packet header) marker can be included after each packet header.

On a fine coding-pass level, several code-block styles may be used to improve error resilience. Particularly, the SEGMARK (segmentation marker) style causes encoding a special symbol at the end of each bitplane. Since the decoder is expecting to find them, the decoding of this symbol confirms the correctness of the decoding of the bit-plane. Alternatively, the combination of both ERTERM (error resilient termination) and RESTART (restart MQ coder) styles provides superior error resilience to that offered by the SEGMARK style. However, the introduced overhead is larger than the cost of the SEGMARK symbol. The RESTART style is used causes that the coder is restarted at the beginning of each coding pass. Thus, when this style

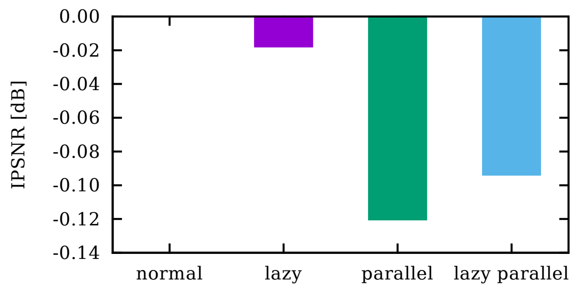

Figure 8: A degradation of the PSNR for the lazy and parallel modes. Compressed at 1.2 bpp.

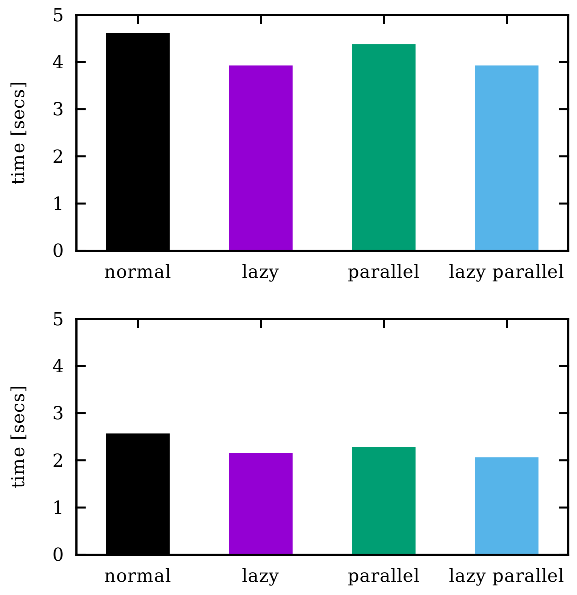

Figure 9: Compression (top) and decompression (bottom) speed for the lazy and parallel modes.

is utilized, every coding pass establishes its own codeword segment. The ERTERM forces the encoder to implement a very specific termination procedure for codeword segments. Therefore, the decoder can detect that an error has occurred in this codeword segment. The above features are detailed in [4]. Moreover, one can find various works, e.g., [17], that investigated the impact of random errors on the JPEG 2000 format.

Since the usage of uncommon code-block styles can cause compatibility issues, we only recommend using the SOP and EPH markers since their overhead is negligible and they are commonly supported on the decoder side.

Reduced computational demands. The often-used way to reduce computational complexity is the BYPASS code-block style (the lazy mode). The usage of this style skips the arithmetic coder for certain coding passes (the binary symbols are stored in raw segments). This speed up the coding with little loss in compression efficiency.

Other opportunities for reducing computational complexity offer the RESTART, RESET, and CAUSAL code-block styles. If all three styles are enabled (the parallel mode), encoder and decoder may process the coding passes within a code-block in parallel. The RESTART restarts the arithmetic coder at the beginning of each coding pass. Further, the RESET (reset context states) reinitializes the context probabilities at each

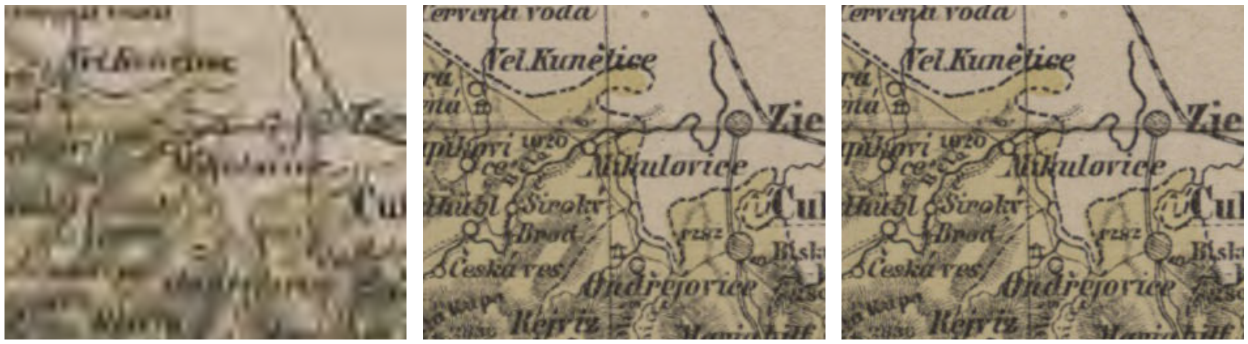

Figure 10: Visual image quality comparison on detail view. Cropped from a larger image. Observe compression artefacts. At first glance, the image with the 1:30 compression ratio is indistinguishable from the original. But when you look at this image in more detail, you may notice some inconspicuous artifacts even here.

coding pass boundary. Finally, the CAUSAL (stripe-causal context formation) ensures that the samples within a given stripe may be coded without any dependence on samples from future stripes. Note that each coding pass follows a stripe-oriented scan.

As investigated in [5], the loss in quality is generally small (about 0.01–0.3 dB) for both the lazy and/or parallel mode. Figures 8 and 9 show the effect of the lazy and parallel modes on compression speed and efficiency for our test image. The lazy mode reduces the compression efficiency significantly less than the parallel mode or its combination. These results clearly match the findings in [5]. Surely, the compression and decomposition speed depends on the codec used. Specifically for the Kakadu, measured values indicate that the lazy mode accelerates the processing time slightly more than the parallel mode. However, the differences are below 20 %.

For the same reasons as in the previous case (compatibility issues), we do not recommend using any of these code-block styles. Besides, they reduce compression performance and their contribution is questionable (support in decoders is poor).

File format. There are several file formats associated with the JPEG 2000 standard – the raw code-stream, JP2, JPX, and other wrapping formats.

The JPEG 2000 code-stream consists of the main header (contains information on image size, tiles, components, etc.) and a stream of segments (mostly packets) containing the compressed image itself. Although the raw code-stream may stand alone, it is not usually used as a file format. If even though the code-stream is saved to a file, the non-standardized extensions .jpc, .j2c or .j2k are commonly used.

However, the JPEG 2000 code-stream is usually embedded in a JP2 file format. The format may be viewed as a wrapper for a code-stream and its major purpose is to associate metadata with the compressed image. The JP2 file format itself is organized as a sequence of boxes. The information that the various boxes carry includes colour space specification, ICC profile, colour palette, capture/display resolution (points per meter), XML metadata, and others. The JP2 files should be given the file extension .jp2, as defined in the standard. Users mostly want to use this format.

The JPX is an extension of JP2 format and provides many enhancements. Such enhancements include more powerful colour space specification architecture, the support for revision history for metadata, possibility of code-stream fragmentation, or ability to combine multiple code-streams (to obtain animation or compositing). The format employs the extension .jpx.

Other options. JPEG 2000 standard and particular encoders offer a lot of additional options. However, these are rarely used and are out of focus in this paper. Specifically, we would like to mention the following

Table 2: Overview of important libraries' parameters. Long dash indicates the inability of the library to insert given segment.

Option	Kakadu	OpenJPEG
enforce lossy compression[1]	`Creversible=no`	`-I`
YCbCr colour transform[2]	`Cycc=yes`	`-mct 1`
tile size	`Stiles=`{*height*, *width*}	`-t` *width*, *height*
image origin	`Sorigin=`{*y*, *x*}	`-d` *x*, *y*
tile origin	`Stile_origin=`{*y*, *x*}	`-T` *x*, *y*
decomposition levels[3]	`Clevels=`*levels*	`-n` *resolutions*
code-block size	`Cblk=`{*height*, *width*}	`-b` *width*, *height*
bitrate for quality layers[4]	`-rate` *rate*, ...	`-r` *ratio*, ...
quality for quality layers[5]	`-slope` *slope*, ...	`-q` *psnr*, ...
define progression order[6]	`Corder=`*order*	`-p` *order*
precinct size[7]	`Cprecincts=`{*height*, *width*}, ...	`-c` *width*, *height*, ...
divide tile into tile-parts[8]	`ORGtparts=`*grouping*	`-TP` *grouping*
insert TLM segments[9]	`ORGgen_tlm=`*N*	—
use PLM segments	—	—
use PPM segments	—	—
include SOP markers	`Cuse_sop=yes`	`-SOP`
include EPH markers	`Cuse_eph=yes`	`-EPH`
enable SEGMARK style	`Cmodes=SEGMARK`	`-M 32`
ERTERM and RESTART styles	`Cmodes=ERTERM|RESTART`	`-M 20`
enable lazy mode	`Cmodes=BYPASS`	`-M 1`
enable parallel mode	`Cmodes=RESET|RESTART|CAUSAL`	`-M 14`
file format[10]	`-o` *file*	`-o` *file*

[1] The reversible compression path uses the 5/3 wavelet, while irreversible compression uses the 9/7 one.
[2] The colour transform is used by default if there are 3 or more components.
[3] The number of *resolutions* is one more than number of *levels*.
[4] The *rate* is expressed in bits/pel. The *ratio* is actual compression factor. Succesive layers are separated by comma.
[5] The *slope* for distortion-length slope, *psnr* for PSNR. Succesive layers are separated by comma.
[6] The *order* is one of the LRCP, RLCP, RPCL, PCRL, or CPRL.
[7] The first record refers to the highest resolution. The last record is used for remaining resolutions.
[8] The *grouping* is one of the R, L, or C.
[9] The *N* is the maximum number of tile-parts for each tile.
[10] The file format is chosen according to the file extension.

options: various regions of interest, progression order change, extensions in Part 2 of the standard, custom quantization factors, component registration, or attaching various metadata. The inquiring reader is referred to [4] for further details.

To better interpret the above guidelines, one needs to gain experience with the dependence of an image distortion on the compression ratio. See Figure 10 to get the basic idea of such a dependence. Observe the compression artefacts in the shape of the spatial wavelets. Considering large-resolution imagery or networked environment, even compression ratios about 1 : 1 000 can be meaningful.

At this point, we overview the usage of the two most frequently used JPEG 2000 software codecs – the OpenJPEG and Kakadu. The OpenJPEG is a reference software, whereas the Kakadu is a commonly used proprietary library. Table 2 reviews their command-line arguments controlling the options discussed above.

3 Preservation Risk?

We cannot provide a clear and eventual answer to the question of whether using the JPEG 2000 format is a long-term preservation risk. However, below we list the five most important aspects to be considered in order to decide whether or not to use it.

Presumably, the biggest disadvantage of JPEG 2000 format is its high complexity. The high complexity consequently increases the resources required for its complete implementation, which in turn increases the price and the barrier to entry for new implementations. Additionally, in order to facilitate its effective use, JPEG 2000 requires an in-depth understanding of the wavelet-based image compression. For example, as mentioned in a report of the British Library in [18], this library had to hire an external consultant to assess their needs and develop an appropriate JPEG 2000 profile when adopting this format. This problem was also mentioned by an archivist and librarian of the Library of Congress in [19].

Another commonly criticized lack of the format is that JPEG 2000 is poorly supported by the majority of popular end-user image editors and viewers, and there is a lack of high-quality tools able to deal with this format. This increases the risk that users will be unable to open the format in their commonly used program.

On the other hand, there is an officially recognized open-source reference codec. At the time of writing this article, this codec is still actively developed. The availability of working open-source software is highly important for all practical purposes (support in open-source software, e.g., ImageMagick conversion tool or IIPImage image server).

Besides the above-mentioned lacks, there are also advantages resulting from the choice of this format. The following one was highlighted in the report of by the Wellcome Digital Library in [15]. Using the JPEG 2000 format offers the ability of exact specification of compressed file size, which does not depend on image content, thereby mitigating concerns about the overall storage requirements. There is even the possibility of specifying multiple bitrates in a single compressed file, which makes sense in conjunction with delivering the images on demand. A common approach for delivering images from an image server is to decode just as much of the image as is needed to create the requested view, and then convert the resulting image to JPEG at the server for delivery to a client.

Another advantage is relatively good error resilience and robustness. For the sake of reference, this aspect is examined, *inter alia*, in [17]. Bit errors (as a result of failures in a data storage device or transmission) in a JPEG 2000 stream create less visual artifacts than other comparable formats, especially JPEG and MPEG. A bit error affects only a single code-block, whose lost usually results in an imperceptible localized blur in the image. As opposed to the other image formats, JPEG 2000 also keeps stable quality regardless of the number of times the image has been recompressed.

As a final remark, we would like to make the following recommendation. If you are serious about deploying JPEG 2000 in your organization, get a good quality commercial codec. Along with this, you get the support of the codec manufacturer, so you get rid of many incompatibility issues with existing open-source solutions. Besides, many open-source codecs are not multi-threaded and thus too slow to practical use.

4 Conclusions

The JPEG 2000 format is quite complex, and therefore sometimes considered as a preservation risk. From the perspective of the user, a lossy JPEG 2000 compression is governed by a number of parameters that control compression speed and rate–distortion trade-off. Their inappropriate adjustment may fairly easily lead to sub-optimal compression performance. This article provided general guidelines for selecting the most appropriate parameters for a specific application. Although there is no universal compression setting, we suggested a recommendation or the commonly used value is given wherever it was meaningful. Furthermore, the article also gave an overview of such parameters in two most commonly used software codecs, which is Kakadu, OpenJPEG. The recommendations are intended for use by involved users, e.g., cultural heritage institutions. The article was also written in a way to be useful to technically unskilled users who have to deal with the format.

The JPEG 2000 format is being used by professional archives and libraries as their long-term storage codec. We are aware of over a dozen of big memory institutions worldwide using this format (list is given, e.g., in [20]). Each of them uses its own JPEG 2000 profiles. As far as we know, the JPEG 2000 is also extensively used in digital cinema, medical imaging, and broadcast studios.

List of abbreviations

DCT: discrete cosine transform; DWT: discrete wavelet transform; CDF: Cohen–Daubechies–Feauveau; PSNR: peak signal-to-noise ratio; IPSNR: improvement of the PSNR; MQ: arithmetic coder employed by JPEG 2000; LRCP: layer, resolution, component, position; RLCP: resolution, layer, component, position; RPCL: resolution, position, component, layer; PCRL: position, component, resolution, layer; CPRL: component, position, resolution, layer; PLT: packet lengths: tile-part; PLM: packet lengths: main header; PPM: packed packet headers: main header; TLM: tile-part lengths: main header; SOP: start of packet; EPH: end of packet header; SEGMARK: segmentation marker; ERTERM: error resilient termination.

Funding

This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II) project IT4Innovations excellence in science (LQ1602), and the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737475.

Authors' contributions

David Barina generated the idea of this work, discussed it with the other author, carried out the main experiments, and wrote the manuscript. Ondrej Klima supervised the project, and approved the final manuscript.

Acknowledgements

The authors would like to thank Pavel Zemcik for providing their feedback on this paper.

References

[1] van der Knijff J (2011) JPEG 2000 for Long-term Preservation: JP2 as a Preservation Format. D-Lib Magazine 17(5/6), DOI doi:10.1045/may2011-vanderknijff

[2] Wallace GK (1991) The JPEG Still Picture Compression Standard. Communications of the ACM 34(4):30–44, DOI 10.1145/103085.103089

[3] Mallat S (2009) A Wavelet Tour of Signal Processing: The Sparse Way. With contributions from Gabriel Peyre, 3rd edn. Academic Press

[4] Taubman DS, Marcellin MW (2004) JPEG2000 Image Compression Fundamentals, Standards and Practice. Springer, DOI 10.1007/978-1-4615-0799-4

[5] Schelkens P, Skodras A, Ebrahimi T (2009) The JPEG 2000 Suite. The Wiley-IS&T Series in Imaging Science and Technology, Wiley, DOI 10.1002/9780470744635

[6] Kozenn B, Kovář MR (ca. 1870–1890) Příruční mapa Markrabství moravského a vévodství slezského. UUID: d01967d0-3bcb-11df-b973-000bdb925259

[7] Chan G (2008) Toward Better Chroma Subsampling. SMPTE Motion Imaging Journal 117(4):39–45, DOI 10.5594/J15100

[8] Taubman D (2002) Software architectures for JPEG2000. In: Proceedings of the IEEE International Conference for Digital Signal Processing, pp 197–200

[9] Drepper U (2007) What Every Programmer Should Know About Memory. Tech. rep.

[10] Varma K, Bell A (2004) JPEG2000–Choices and Tradeoffs for Encoders. IEEE Signal Processing Magazine 21(6):70–75, DOI 10.1109/MSP.2004.1359144

[11] Barina D, Klima O, Zemcik P (2016) Single-Loop Architecture for JPEG 2000. In: International Conference on Image and Signal Processing (ICISP), Springer, Lecture Notes in Computer Science (LNCS), vol 9680, pp 346–355, DOI 10.1007/978-3-319-33618-3_35

[12] Buckley R (2008) JPEG 2000 - a Practical Digital Preservation Standard? Tech. rep.

[13] ITU-T (2000) Recommendation T.800. Information technology – JPEG 2000 image coding system: Core coding system

[14] Buckley R (2013) Using Lossy JPEG 2000 Compression For Archival Master Files. Tech. rep.

[15] Buckley R (2009) JPEG 2000 as a Preservation and Access Format for the Wellcome Trust Digital Library. Tech. rep.

[16] Buckley R, Sam R (2006) JPEG 2000 Profile for the National Digital Newspaper Program. Tech. rep.

[17] Buonora P, Liberati F (2008) A Format for Digital Preservation of Images: A Study on JPEG 2000 File Robustness. D-Lib Magazine 14(7/8), DOI 10.1045/july2008-buonora

[18] Wheatley P, May P, Pennock M, Whibley S (2015) Preservation assessment: JP2 format. British Library Digital Preservation Team

[19] LeFurgy B (2013) Is JPEG-2000 a preservation risk? The Signal, Library of Congress

[20] Ostráková N (2018) JPEG 2000 jako archivní formát obrazových dat. Knihovna: knihovnická revue 29(1):5–26, in Czech