Multimedia Tools and Applications, 26, 27–57, 2005
© 2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

Performance Analysis of the JPEG 2000 Image Coding Standard

HONG MAN hman@stevens-tech.edu
Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030-1503, USA

ALEN DOCEF adocef@vcu.edu
Department of Electrical Engineering, Virginia Commonwealth University, Richmond, VA 23284-3072, USA

FAOUZI KOSSENTINI
UB Video, Inc., Suite 203, 1038-1040 Hamilton Street, Vancouver, BC, Canada V6B 2R9

Abstract. Some of the major objectives of the JPEG 2000 still image coding standard were compression and memory efficiency, lossy to lossless coding, support for continuous-tone to bi-level images, error resilience, and random access to regions of interest. This paper will provide readers with some insight on various features and functionalities supported by a baseline JPEG 2000-compliant codec. Three JPEG 2000 software implementations (Kakadu, JasPer, JJ2000) are compared with several other codecs, including JPEG, JBIG, JPEG-LS, MPEG-4 VTC and H.264 intra coding. This study can serve as a guideline for users to estimate the effectiveness of JPEG 2000 for various applications, and to select optimal parameters according to specific application requirements.

Keywords: image compression, JPEG-2000

1. Introduction

Since the release of the Joint Photographic Experts Group (JPEG) still image coding standard in 1994 [7], it has been widely adopted in applications involving digital communication and storage of still images and graphics. Motivated by the evolution of image coding technology and by an increasing field of applications, the JPEG committee initiated a new project in 1997 to develop the next generation still image coding standard. The joint effort of the International Organization for Standardization (ISO) and the International Telecommunication Union (ITU-T), resulted in the JPEG 2000 International Standard [10], published in December 2000.

The original Call for Contributions for the JPEG 2000 standardization effort [8] identified a set of coding features believed to be vital to many existing and emerging image applications. These were translated into goals for the new standard, as shown below.

- The system should offer excellent compression performance at very low bit rates, typically 0.25 bits-per-pixel (bpp) or less.

- The system should be able to compress both continuous-tone and bi-level images with similar system resources.
- The system should provide lossy to lossless compression by means of a progressive coding process.
- The system should be able to perform progressive coding in terms of both pixel accuracy and spatial resolution.
- The system should produce code streams that are robust to channel errors.
- The system should allow random access to and processing of certain parts of the image such as regions of interest.

Various techniques have been proposed to address these requirements. The standard eventually converged to a baseline coding system that achieves a good balance in supporting the desired features.

The purpose of this paper is to provide readers with some insight on the coding structure of JPEG 2000, its functional parameters, and their effect on coding performance. The codec is also compared to several other standard codecs, including JPEG, JBIG, JPEG-LS, MPEG-4 VTC and H.264 intra coding. The performance measures we take into consideration when comparing the codecs are compression efficiency, lossless coding, bi-level image coding, computational complexity, handling of large images, progressive coding, and error resilience. Other aspects should be considered when evaluating an image codec for a particular application, including perceptual image quality, scalability, rate control precision, and intellectual property rights. These issues are beyond the scope of our work.

This study can serve as a guideline for the evaluation of the effectiveness of JPEG 2000 for various applications, and for the selection of effective options and parameter settings according to particular application requirements. Comparative analyses can be found in the literature [14, 18, 20], but our study attempts to be more extensive by considering a wide variety of images, a comprehensive set of performance measures, and all the codecs and implementations currently available. It is not our intention to discuss the detailed coding algorithm and its implementations in this paper. For such information, the reader should refer to a tutorial on the coding engine by Taubman [20], a system level introduction by Christopoulos et al. [3], and the standard text [10].

The remainder of this manuscript is organized as follows. Section 2 provides a detailed description of the JPEG 2000 coding algorithm and its parameters and data structures. Alternate algorithms are briefly discussed in Section 3. The experimental setup and results are given in Section 4, and conclusions are drawn in Section 5.

2. The JPEG 2000 coding algorithm

JPEG 2000, like the other codecs tested in this paper, is essentially a transform coder, which consists of three stages: image transform, quantization and entropy coding. An image transform is used to achieve image data decorrelation. An efficient transformation yields a representation of the image data where the energy is concentrated in a small number of transform coefficients. Transform coefficients are then quantized to a finite number of quantization levels. It is during quantization that intentional loss of information occurs and most of the compression gain is achieved. Finally, the quantized coefficients are scanned and encoded into a bit stream using an entropy coder. An image decoder performs the inverse operations to obtain a reconstructed image.

The coding engine of the JPEG 2000 standard is a coding algorithm derived from the Embedded Block Coding with Optimal Truncation (EBCOT) technique proposed by Taubman [20]. Detailed descriptions of the codec are given in, among others [1, 17, 22]. In this section, we briefly describe the basic concepts, parameters, and structures of the coding algorithm. Although the JPEG 2000 standard, like many other coding standards, only defines the decoder operations, it also provides some informative description of the encoder implementation. Our discussion is written primarily from the encoder point of view.

In our tests, we used three software implementations of the JPEG 2000 codec: the JasPer implementation [24], the Kakadu implementation [21], and the JJ2000 implementation [5].

2.1. The image transform

The JPEG 2000 algorithm is based on the Discrete Wavelet Transform (DWT). It is therefore not compatible with the JPEG coding algorithm, which uses the two-dimensional (2-D) Discrete Cosine Transform (DCT). Studies have shown [2] that a well designed DWT may have a moderate gain over DCT in the sense of data decorrelation. More importantly, a significant improvement in performance is achieved by applying the transform to the whole image or large blocks thereof instead of small image blocks (JPEG uses 8×8-pixel blocks). The drawback of this approach is its increased computational complexity. The selection of DWT at the beginning of the JPEG 2000 project essentially determined the coding structure of this new standard.

The 2-D DWT is usually implemented through iterative 2-D subband decomposition, as seen in figure 2. First, the image is decomposed into four subbands, LL_N, LH_N, HL_N, and HH_N. The same 2-D subband decomposition is then applied to the lowest frequency subband (LL_N) to obtain subbands LL_{N-1}, LH_{N-1}, HL_{N-1}, and HH_{N-1}. The process is repeated N times, for an N-level decomposition. Commonly, the number of decomposition levels is around $N = 5$. This subband decomposition structure is called the *Mallat* (or *dyadic*, or *pyramid*) decomposition. This is the only decomposition structure supported by Part 1 of the standard. The inverse DWT transformation consists in N 2-D subband synthesis operations applied starting at the lowest resolution.

A one-level 2-D subband decomposition is usually achieved by applying a one-dimensional (1-D) two-band decomposition to all the rows (or columns) of a 2-D array and then to all the columns (or rows) of the resulting array. Therefore, the decomposition generates four subbands, as shown in figure 2. Each 1-D subband decomposition consists in filtering the input sequence with a low-pass/high-pass filter bank and downsampling the two filtered sequences by a factor of two. JPEG 2000 specifies two sets of 1-D filter banks. The reversible 5/3 filter bank performs a reversible integer-to-integer transformation which can be used to achieve lossless coding. The irreversible 9/7 filter bank performs a real-to-real transformation which is reversible in infinite precision but irreversible in finite precision. The resulting DWT achieves better energy compaction and is used in lossy coding.

In addition to the DWT for spatial decorrelation, the standard also defines two sets of component transforms for multi-component images, such as color images. Again, an irreversible component transformation (ICT) is defined for lossy coding and a reversible component transformation (RCT) is defined for lossless coding.

At the very beginning of the encoding process, a DC level shift is applied to each image component. For an image component of bit depth B, the quantity 2^{B-1} is subtracted from all the pixel values, in an attempt to obtain a zero-mean distribution. This level shift is compensated for at the decoder.

2.2. *Quantization*

To quantize DWT coefficients, JPEG 2000 uses a uniform scalar quantizer with a dead zone around zero. This quantizer has been chosen mostly because of its simplicity. It can easily be implemented as a rounding operation. Each DWT coefficient $a_b(u, v)$ in subband b is quantized to the integer $q_b(u, v)$ using the formula

$$q_b(u, v) = sign(a_b(u, v)) \left\lfloor \frac{|a_b(u, v)|}{\Delta_b} \right\rfloor,$$

where Δ_b is the quantization step, which can vary from one subband to another and is encoded into the bit stream. The standard does not specify how the quantization step sizes Δ_b are chosen, and different implementations can use various strategies. The decoder performs inverse quantization, described by the equation

$$\hat{a}_b(u, v) = \begin{cases} 0, & \text{if } q_b(u, v) = 0 \\ sign(q_b(u, v))(|q_b(u, v)| + 0.5)\Delta_b, & \text{if } q_b(u, v) \neq 0. \end{cases}$$

In lossless coding, coefficients of reversible transforms are not quantized, or they can be thought as quantized by a step size of one. In lossy compression, the quantization step sizes control the final size of the compressed data file, and thus the compression ratio. The larger the step sizes are, the smaller the compressed file size will be. Therefore, selecting the step sizes is a rate control issue and will be discussed in a later section.

In general, a well-designed quantizer minimizes the quantization distortion at certain bit rates, using an objective measure of the distortion such as the peak-signal-to-noise ratio (PSNR). For a wide class of signals, the same quantization step must be chosen for all subbands to optimize the PSNR. However, objective distortion measures don't always truly reflect the perceptual quality. In applications where perception is more important than accuracy in the decoded image, the encoder can use *visual weighting*. The technique is similar to the use of a quantization matrix in JPEG and it consists in choosing a quantization step Δ_b according to the average contrast sensitivity of the human visual system in that frequency band. Thus, smaller quantization steps will be used for lower-frequency bands. This adjustment only affects encoder functionality, and is transparent to the decoder.

2.3. Entropy coding

The major differences between various wavelet-based image coding algorithms are mostly in the lossless entropy coding stage. This is the step that affects their different coding performances and features to the greatest extent.

2.3.1. Bit-plane coding. To encode quantized DWT coefficients, JPEG 2000 uses a block based bit-plane coding method with multiple passes per bit-plane. Each subband is partitioned into a set of non-overlapping rectangular *code blocks* with a fixed size. Each block is then encoded independently of other blocks. Within each block, the coding starts from the most significant non-zero bit-plane and proceeds to the least significant bit-plane. In each bit-plane, three coding passes are performed, namely the *significance propagation pass*, the *magnitude refinement pass* and the *cleanup pass*. Each coding pass scans the block according to a scanning pattern, and generates coding symbols using a set of pre-defined rules. The scanning pattern, shown in figure 1, has been chosen to facilitate efficient software and hardware implementations.

During the significance propagation pass, the encoder identifies the block coefficients which have been zero (or *insignificant*) at higher bit-planes but have non-zero (or *significant*) neighbors. It is assumed that these coefficients are more likely to become significant in the current bit-plane. For each tested coefficient, the test result is encoded into the bit stream (1 if significant, 0 if insignificant). If a new significant coefficient is detected, its sign bit is encoded immediately.

The magnitude refinement pass processes coefficients which have been found significant in previous bit-planes. For each such coefficient, the magnitude bit in the current plane is encoded into the bit stream.

The cleanup pass tests all the insignificant coefficients which have not already been tested in the significance propagation pass in the current bit-plane. It is expected that these coefficients are less likely to become significant because they do not have any significant neighbor. Therefore, to reduce the number of symbols generated in this pass, test results are run-length encoded when four consecutive coefficients in a scan pattern column are all insignificant. Once a significant coefficient is identified, its sign bit is also encoded immediately.

Each coefficient in a code block will be coded once and only once in each bit-plane. The symbols generated by these three passes are usually passed through a binary adaptive arithmetic coder with context modeling. Specific context models are defined for the coding of each pass and for the coding of sign bits. Each context determines which adaptive probability model will be used in the arithmetic coding of the current symbol. The contexts are selected based on the significance of the eight or four connected neighboring coefficients. To simplify context calculation and achieve reliable probability estimation, the number of contexts is kept small. More specifically, the significance propagation pass uses nine contexts, the magnitude refinement pass uses three contexts, the cleanup pass uses the nine significance propagation contexts plus one extra context for run-length coding, and the sign bit coding uses five contexts. JPEG 2000 uses a binary adaptive arithmetic coder called the *MQ* coder, designed to reduce computational complexity. It is closely related to the well-known *QM*

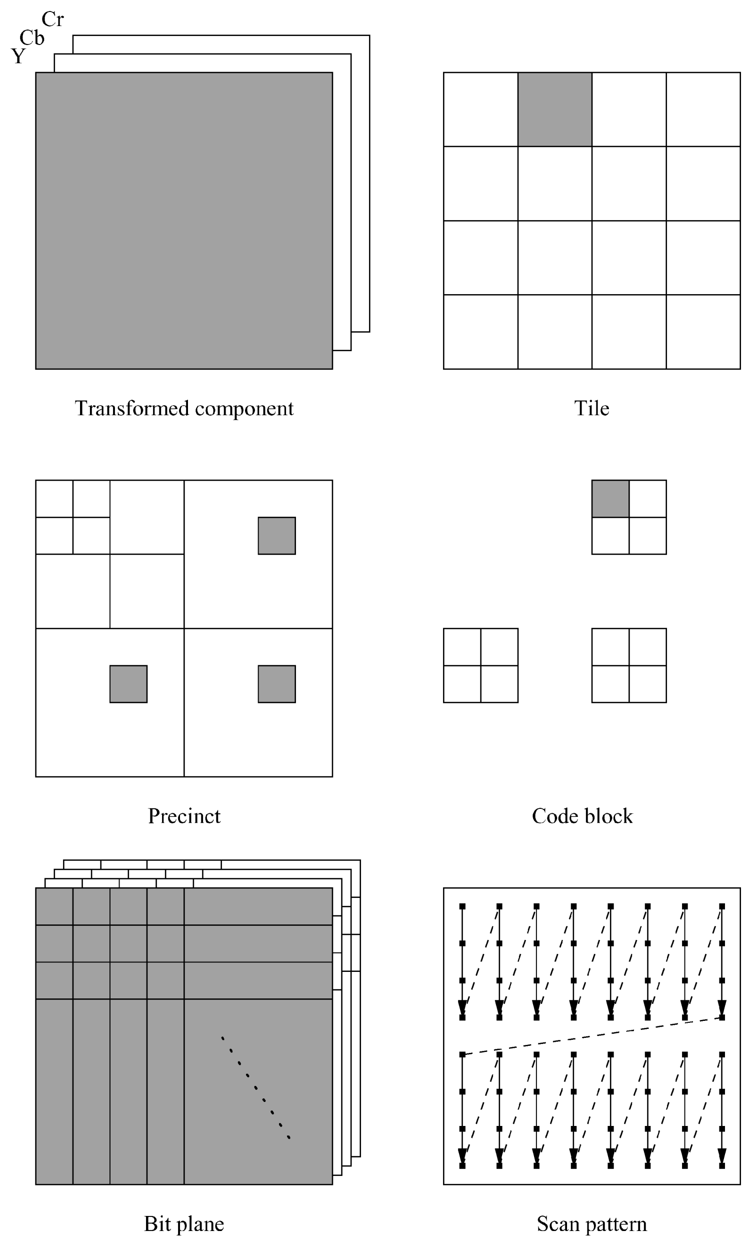


Figure 1. Components of the JPEG 2000 image data structure.

and an offspring of the Q coder. Arithmetic coding is always initialized at the beginning of a code block and can be optionally reinitialized at the beginning of a coding pass.

JPEG 2000 also defines a *selective arithmetic coding bypass* or *lazy coding* mode in which, after the first four bit planes, significance propagation and magnitude refinement symbols bypass arithmetic coding. The purpose of this coding mode is to further reduce computational complexity without significant loss in compression performance.

2.3.2. Rate control. One rough method for rate control has been mentioned in the context of quantization. However, the multipass bit-plane coding approach described above allows for rate control with better precision and reliability.

A distinguishing feature of JPEG 2000 is its post-compression rate-distortion (PCRD) optimization [22], or optimal bit stream truncation. Its purpose is to determine the optimal bit allocation for each code block under the constraint of a target bit rate.

During quantization, a very fine quantization step size Δ_b is used for all the coefficients inside a subband. Encoding of each code block results in a bit stream which represents all the quantized coefficients to a very fine scale. For most useful compression ratios, it is not possible to send all these streams in their entirety. Therefore, these streams are subject to truncation in order to meet the overall target bit rate. PCRD attempts to choose the truncation points in a way that minimizes the coding distortion for a target bit rate. The possible truncation points in a bit stream are at the end of each coding pass. If a bit stream of a certain block is truncated at a bit-plane that is N bit-planes higher than the least significant bit plane, the effective quantization step of this block is $2^N \Delta_b$.

During the coding process, whenever a coding pass is completed, the bit rate consumption and the reduction in distortion due to this coding pass are calculated and recorded. When the coding of a whole block is completed, the rates and distortions of all the passes are recorded in a rate-distortion (R-D) table. Similar tables are generated for all code blocks. PCRD optimization uses a Lagrangian optimization method to determine which rate-distortion pair is to be used for each code block. Details for the optimization routine are given in [22], pp. 339–348, and are beyond the scope of this paper.

An *embedded code stream* is a code stream that can be decoded as a whole, or it can be truncated and decoded at various bit rates lower than the original bit rate. Therefore, the lower bit rate streams can be seen as embedded in a higher bit rate stream. This scalability feature can be helpful in applications involving multiple bandwidth channels or multiple decoding platforms. In this context, although the complete code stream is optimized in the R-D sense, an arbitrarily truncated stream may not be optimal at its reduced bit rate. To address this problem, EBCOT supports the concept of *quality layers*. During PCRD optimization, instead of performing one R-D optimization for a single target rate, the coder can perform a series of R-D optimizations from lower rates to higher rates. Each new optimization is built upon the previous optimal bit stream truncations, and each optimization generate a quality layer that will be appended to previous layers to form the final code stream. Therefore, any truncation of the code stream at the end of a quality layer will always be optimal in the R-D sense. The number of bits corresponding to each data block included in a layer is encoded in the code stream as header information.

2.3.3. Regions of interest. Many applications may require that some areas within an image (or *regions of interest*, ROI) be encoded with higher accuracy than the rest of the image. Therefore, at a certain total bit rate, the encoder should allocate more bits for ROI pixels and less bits for for non-ROI (or background) pixels. This feature usually involves an *ROI mask* and a shift *scale* s. The ROI mask is a binary map defining an arbitrary-shape region of interest. A set of smaller ROI masks are calculated for each subband according to the original mask. The shift scale s specifies how much the ROIs will be emphasized. It is used to shift the quantization indices of all ROI subband coefficients to higher bit-planes, and effectively amplify their magnitude by a factor of 2^s. This shift will be corrected at the decoder. In Part 1 of the standard, the ROI implementation is based on the Maxshift method [22], in which the shift scale is always larger than the largest number of magnitude bit-planes of all the background regions. As a result, after the ROI shift, the least significant bit-plane of the ROIs will still be higher than the most significant bit-plane of the background regions. This shift scale will have to be coded into the code stream and sent to the decoder. Entropy coding of the shifted subband coefficients is performed as usual. Since the decoder is using bit-plane decoding, it will always complete the decoding of all ROI coefficients before it can reach the background coefficients. Therefore the decoder can determine the ROI mask as the set of significant coefficients after s bit planes have been decoded. Thus the ROI mask does not have to be transmitted.

2.4. The JPEG 2000 data structure

The JPEG 2000 defines a set of data structures that standardizes the access and reference to all the data involved. One set of structures is related to the processing of image data, the other is related to the formation of the code stream.

2.4.1. The image data structure. A diagram showing the image data structure is shown in figure 1. An *image* refers to the input of the encoder and the output of the decoder. The associated data structure is described below, the components being listed in decreasing order of their size.

Component: An image is first separated into one or more components. A component can be an arbitrary 2-D rectangular array of samples. The most common image components are the color components representing the three color (R, G, B) planes of an image. The inter-component transform defined in the standard is mainly used with RGB images. However, other decompositions are possible for multi-component images such as radar images (in-phase and quadrature) or printer output images (the CMYK color space). Components do not necessarily have the same size, nor the same bit depth.

Tile: Each component is divided into one or several tiles, which are then encoded independently. All tiles are the same size, except tiles at the boundaries. The purpose of tiling is to allow for the coding of very large images by reducing memory consumption and speeding up the coding process. Tiling may cause a slight decrease in compression efficiency, since the ability to exploit spatial redundancy is reduced when small tiles are used. Also, at low bit rates, blocking artifacts may be visible at tile boundaries.

Subband: Each tile in a particular component, is encoded using the JPEG 2000 DWT coding algorithm. The tile is input to the wavelet transform, which generates a set of

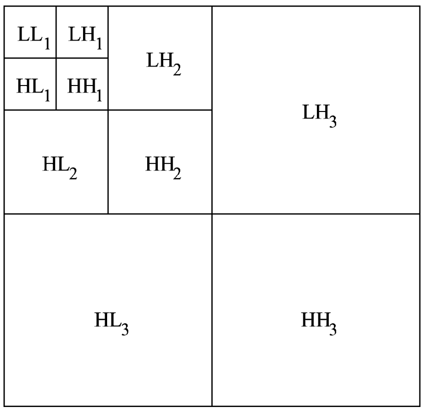


Figure 2. A three-level ($N = 3$) 2-D DWT decomposition.

subbands representing different spatial frequency components. Due to subsampling in the wavelet transform, subband sizes decrease by a factor of four at each subband decomposition level.

Precinct: At a particular resolution level except the level 0, a precinct contains a group of code blocks that cover the same rectangular area in all three subbands. At level 0, a precinct is just a group of code blocks in the subband LL_1. Therefore a precinct represents all the information in a particular image region at a certain resolution level. It is then reasonable to suggest that such information should be packed together in the final code stream. In the code stream, bits are organized by precincts, instead of subbands.

Code block: After the quantization of all subband coefficients, each subband is partitioned into non-overlapping rectangular code blocks. These code blocks are confined within a certain subband, and have the same size except at boundaries of the subband. The maximum size of a code block is 64×64. They are input to the EBCOT coding engine. Each code block is encoded independently, and produces a bit stream to be included in the final code stream.

2.4.2. The code stream data structure. The code stream refers to the output of the encoder and the input of the decoder. Its associated data structure, sketched in figure 3, includes the following components, in increasing order of their size.

Encoded code block: After a code block is encoded using the EBCOT coding engine, a bit stream results. This is the smallest independent unit in the JPEG 2000 code stream.

Packet: Packets are the basic data units in the final code stream. A packet contains all the encoded code blocks from a specific precinct in a tile at a quality layer. The length of a packet can be affected by the size of the precinct and the number of layers. Packets may certainly have different lengths, but they are always aligned at 8-bit boundaries in the final code stream. Each packet has a header indicating its content and related parameters.

Layer: We have introduced the concept of quality layer, which provides finer code stream truncation points with R-D optimization. In order to preserve the embedded feature of the

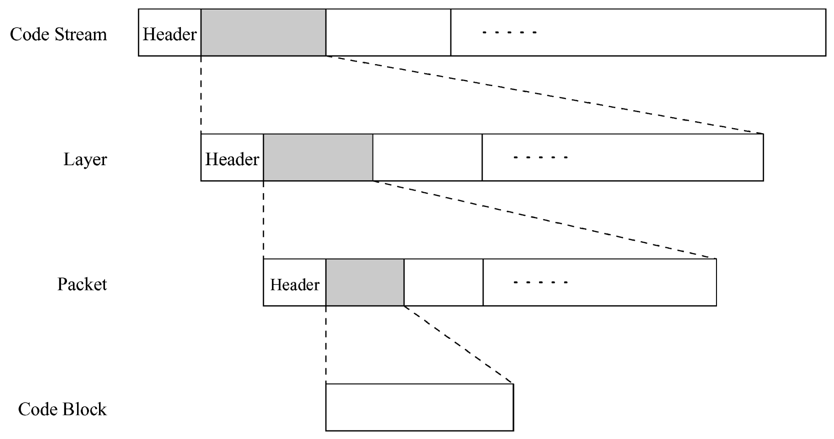


Figure 3. Components of the JPEG 2000 code stream structure.

layered coding mechanism, the final code stream should be organized by quality layers. The lower layers should be at the front of the code stream and the higher layers should be at the end. When the code stream is truncated at the decoder, the lower quality layers will be used to reconstruct the image.

Resolution level: The resolution level is closely related to the decomposition level in the discrete wavelet transform in figure 2. The four subbands, LL_l, LH_l, HL_l, and HH_l at each decomposition level l represent a particular resolution. For example, LL_1, LH_1, HL_1, and HH_1 can be used to reconstruct the image at its original resolution. Also, LL_2, LH_2, HL_2, and HH_2 can be used to reconstruct LL_1, which is the original image at $\frac{1}{4}$ resolution. The resolution levels are defined as follows: LL_N belongs to resolution 0, LH_N, HL_N and HH_N belong to resolution level 1, LH_{N-l}, HL_{N-l} and HH_{N-l} belong to resolution level $l + 1$, and finally LH_1, HL_1, and HH_1 belong to resolution level N.

2.4.3. Progression order. Because the code stream can be arbitrarily truncated at the decoder, the order of packets in the code stream will affect the decoding performance. The standard defines a set of progression orders, which can be easily implemented through re-ordering the packets in the code stream. The LRCP progression first encodes all the Precincts in a component, then all Components at the current resolution level, then all the Resolution levels for the current quality layer and finally sequences through all quality Layers. This procedure is implemented using four nested loops, where the precinct loop is the innermost and the quality layer loop is the outermost. Four additional progression orders are defined: RLCP, RPCL, PCRL, and CPRL. Notice that the LRCP order is accuracy progressive, RLCP and RPCL are resolution progressive, while PCRL and CPRL are spatially progressive.

2.5. Complexity analysis

The most complex coding tasks are clearly the DWT and block coding. Three complexity measures are discussed: memory size, affecting system cost, and memory bandwidth and number of computations, affecting execution speed. The implementation details described in this section are discussed in in Chapter 17 of [22]. We assume here that a single tile is used, covering the entire image.

DWT memory size: When memory size is not an important constraint, the entire image can be stored in memory, and the DWT can be computed in place. A conservative estimate of the maximum bit depth of various subbands is 16 bits. Then, for an image of size N_1 rows by N_2 columns, $2N_1N_2$ bytes of memory are necessary for the DWT step. The same amount of memory is required at the decoder. In many applications, such as satellite imaging, high-quality scanning and printing, the image size is very large and the entire image cannot be stored. A memory-saving pipelining technique can be used in such applications. Byte-size image pixels are presented to the encoder in a continuous stream and are "consumed" by the encoder. If the 9/7 filter bank is used, the DWT stage N buffers nine input rows. The buffered pixels are sufficient to compute a row for each of the LL_N, LH_N, HL_N, and HH_N subbands. The computed rows are not stored: the LL_N line is piped into the next DWT level, while the lines corresponding to LH_N, HL_N, and HH_N are fed to the EBCOT block coder. Therefore, the memory size required for the first DWT level is $9N_2$ bytes. The process continues at level $N - 1$, where $9N_2/2$ coefficients are buffered. These coefficients are 16-bit values though, so $18N_2/2$ bytes of memory are required. The total necessary memory size for the DWT is then $9N_2(1 + 1 + 1/2 + 1/4 + \cdots) \simeq 27N_2$ bytes. Memory size is reduced by a factor of $2N_1/27$ when pipelining is used. A similar pipelining technique can be used at the decoder.

DWT memory bandwidth: This parameter is measured as the average number of byte read or write operations per original image pixel. If the pipelining technique described above is used, the memory bandwidth is 9.2 byte transactions per sample. If M subband lines are computed at a time instead of just one, the memory bandwidth can be reduced at the expense of a higher memory size. For example, for $M = 8$, the memory bandwidth is reduced to 4.1 while the memory increases by a factor of 2.5 ([22], p. 681).

Block coding memory size and bandwidth: Since the DWT outputs data in pixel order and block coding processes data in bit-plane order, the complete block must be buffered before being encoded. For blocks of maximum size (64×64), it can be shown that as many as $3N_2/64$ blocks are simultaneously encoded at any time if pipelining is used. Therefore, $192N_2$ quantized coefficients need to be stored at any given time. The memory bandwidth is 3 byte transactions per sample.

Entropy coding computational complexity: Unlike the DWT computations, which are highly regular and repetitive and consist of multiply-and-accumulate operations, entropy coding operations include many comparisons and branches. These can disrupt the CPU execution pipeline and significantly increase execution time. The MQ coder must be carefully implemented such that the most probable execution paths have the smallest number of branches, thus optimizing average pipeline utilization. Another technique used to speed up entropy coding is state broadcasting. This technique reduces the number of memory accesses necessary for context computation. Every time a coefficient becomes significant, the contexts of the neighboring coefficients are updated accordingly and those contexts need not be read again from memory. Memory bandwidth is reduced because at typical bit rates most coefficients never become significant.

3. Description of the alternate codecs

Before proceeding to performance comparison, we briefly describe the alternate coding algorithms and comment on the particular implementations used in the experiments.

JPEG: JPEG was the first popular lossy image compression standard [7]. It is a block-based transform coding method. An input image is first partitioned into 8×8 blocks. A 2-D 8×8 DCT is then applied to each block, resulting in 64 DCT coefficients. The coefficients are quantized using a uniform quantizer, with finer quantization steps for low frequency coefficients and coarser for high frequency coefficients. The DC coefficients for all the blocks in the image are encoded using a 2-D DPCM. The AC coefficients within each block are zigzag scanned and run-length coded. The resulting DC and AC components are finally variable-length encoded. In our tests, the JPEG implementation from the Independent JPEG Group [23] is used.

JPEG-LS: JPEG-LS is the new lossless image coding standard adopted by the JPEG committee [9, 27]. It is a low-complexity algorithm based on DPCM, two-dimensional prediction, and Rice-Golomb entropy coding. Pixels are processed in conventional raster order. If a pixel resides in a smooth area, a run-length coding scheme is applied. Otherwise, a median edge detection (MED) predictor is used to form the prediction. Run-length symbols and prediction errors are encoded using adaptive Rice-Golomb coding with elaborate context modeling. The codec also supports near-lossless coding by quantizing the prediction errors. The pixel coding error is equal to at most half the quantization step. In our tests, we used the UBC implementation of the JPEG-LS codec [25].

JBIG: The Joint Binary Image Expert Group (JBIG) is responsible for standardization efforts involving bi-level images. The first version of the standard, or JBIG1, was released in 1993 [6]. Its objective was to provide improved lossless compression performance for business-type documents and binary halftone images. It contains a lossless compression algorithm and a progressive coding algorithm. The compression algorithm is essentially an adaptive arithmetic coding with context modeling. The arithmetic coding algorithm is implemented using the *QM coder*. The second generation JBIG standard, or JBIG2, was released in 2001 [11]. While JBIG1 was intended to code generic bi-level images, JBIG2 defines a set of coding methods to code different type of image regions separately. In JBIG2, an image is first segmented into regions that fall into three categories. *Symbol regions*, containing mostly text data, are encoded using a 2-D dictionary-based coding method with character-based pattern matching techniques. *Halftone regions*, containing halftone images, are encoded with another dictionary-based coding method, using halftone templates. *Generic regions*, which can not be classified into one of the two previous categories, are encoded using a bitmap encoder like JBIG1. In our tests, the JBIG-kit [15] by Markus Kuhn and the the UBC JBIG2 implementation [26] have been used.

MPEG-4 VTC: MPEG-4 was adopted by the ISO/IEC Moving Picture Experts Group and became an international standard in 2001 [12]. It defines a video coding standard, an audio coding standard and specifications for the related system. The video coding standard contains a separate tool for still texture image coding, i.e. visual texture coding (VTC). Like JPEG 2000, VTC is based on the discrete wavelet transform. To achieve various levels of spatial and quality scalability, three quantization modes are supported (single, multiple, and bi-level) as well as two scanning modes of the transform coefficients (tree-depth and subband-by-subband). The lowest frequency (DC) subband is encoded using 2-D DPCM, and all higher frequency (AC) subbands are encoded using a zero-tree coding method [19]. Zero-tree coding exploits the fact that, after quantization, a large percentage of transform coefficients are zero. Moreover, if a wavelet coefficient at a certain scale is zero, then its child coefficients in the wavelet tree are very likely to be zero. The zero-tree structure used by this coding technique is a very efficient representation of the location of the zero coefficients. Finally, the DC DPCM symbols, the zero-tree symbols, and the non-zero transform coefficients are encoded using an adaptive binary arithmetic coder. In our tests, the Microsoft MPEG-4 implementation [16] has been used.

H.264-Intra: The H.264 standard currently developed jointly by the ISO and ITU-T [13, 28] is intended to provide enhanced video coding performance compared to both the ISO MPEG-4 and the ITU-T H.263v2 video codecs. The codec provides bit rate savings of up to 50%, consistently high video quality in a wide range of bit rates, a low-delay mode for real-time communications applications, error resilience for handling packet loss bit errors, and flexible packetization and information priority control. A good analysis of the codec's video coding performance is given in [29]. Coding of Intra pictures relies on spatial prediction, an integer transform, quantization, and variable length coding. Spatial redundancy in the image is exploited by prediction of 4×4-pixel blocks or 16×16-pixel macroblocks from surrounding previously encoded blocks or macroblocks, typically the ones located on top and to the left. The prediction error is transformed using an integer 4×4-pixel transform that is an approximation of the DCT. The transform coefficients within a macroblock are quantized using a uniform quantizer with a quantization step that can vary from one macroblock to another. Finally, the quantized coefficients are encoded using a Universal Variable Length Code. In our tests, we used the reference software implementation of the H.264 codec employed by the ISO/ITU-T Joint Video Team [4].

4. Experimental results

4.1. Test environment

All tests have been conducted on a SUN Ultra10 Model 440 workstation with a 440 MHz UltraSPARC-IIi processor, 384 MB DRAM, 2 MB L2 cache, running the SunOS 5.8 operating system. All c/c++ source code has been compiled with the GNU `gcc` compiler version 2.95.2 with "-O2" optimization. The java source code has been compiled with the Solaris JDK 1.2.1 `javac` compiler. For all codecs, coding performance was measured by the peak signal-to-noise ratio (PSNR), execution time as measured by the UNIX `time` command, and memory consumption tracked with the `mpatrol` tool (version 1.4.8) for c/c++ programs and the `hprof` tool for java programs.

Although considerable effort has been made to ensure fairness of the tests, in order to correctly interpret the test results, several issues need to be mentioned.

- Only Kakadu, JasPer and JJ2000 allow the user to specify a target bit rate. Rate control for the other lossy coders is a lot less precise.
- Because the PSNR is used to measure the compression performance, the default visual weighting configurations in JPEG and Kakadu have been disabled, and constant quantization steps have been used.
- Both MPEG-4 VTC and H.264-intra codecs are embedded in their video coding structures, so the computing time and memory usage of their image coding algorithms are very likely overestimated.
- Since both MPEG-4 VTC and H.264-intra codecs can only take YUV input with 4:2:0 sampling, the the test images had to be modified accordingly.
- The memory usage provided by `mpatrol` indicates the peak allocation, while `hprof` only provides the total memory allocation.
- The current JJ2000 decoder contains a bug that prevents correct decoding of some large images in the Solaris java VM.
- It appears that the current H.264 decoder has some problems with irregular image sizes.

The test images used in the experiments are shown in figures 4 and 5. They are:

- gray-level images (8 bits/pixel): LENA (512×512), GOLD HILL (512×512), ULTRASOUND (512×512), CHART (1688×1688), CAFE (2048×2560);
- color images (24 bits/pixel): BIKE (2048×2560) and WOMAN (2048×2560);
- bi-level images (1 bit/pixel): CCIT1 (3504×4750) and CCIT2 (3072×4352).

4.2. Compression efficiency

Lossy compression. Tables 1 and 2 and figure 6 present the results of the compression performance comparison between the various lossy codecs. Most of these codecs have been tested using their optimal compression settings. A rough time profiling of major encoder and decoder components has been performed for both Kakadu and JasPer, and results are shown in Tables 5(a) and (b). Several comments on the results follow.

All three JPEG 2000 implementations achieve effectively the same PSNR performance. However, the different implementations have significantly different speed and memory consumption. JasPer is an intuitive straightforward implementation. Its memory usage increases almost linearly with the image size. Kakadu employs a series of algorithmic optimizations as discussed in Section 2.5. Therefore it is at least three times faster and consumes about 10 times less memory than the other implementations. Some of these optimizations are not limited to the JPEG 2000 standard, and they could be used in other codecs. JJ2000 runs relatively slow with small images, which is somewhat expected because of the Java VM overhead. The time profiling also indicates that the wavelet transform is generally the most computationally intensive component in the coding algorithm.

JPEG 2000 implementations outperform JPEG by 1...3 dB in PSNR. The difference is relatively consistent across various bit rates. JPEG is about three times faster than Kakadu, and about ten times faster than JasPer. The JPEG encoder implementation saves the whole

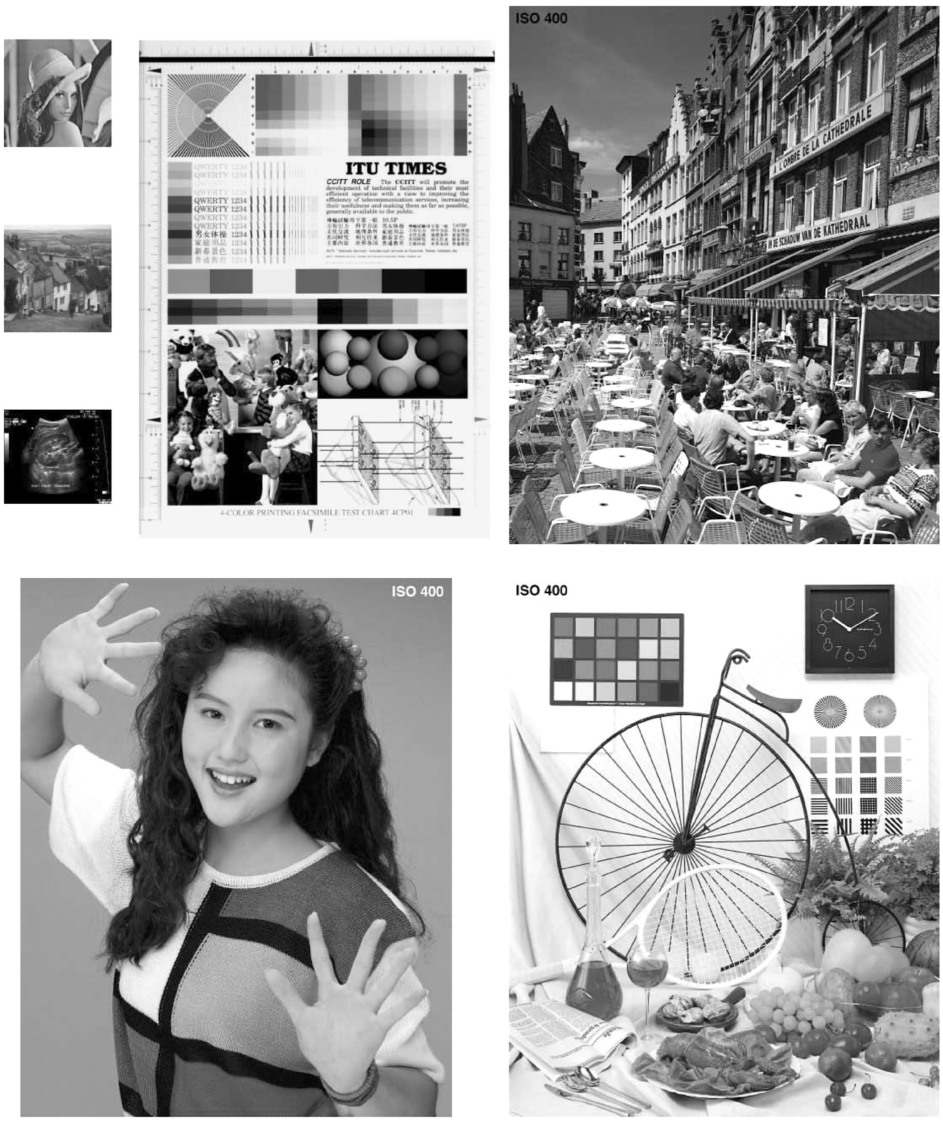

ISO 400
ITU TIMES
4-COLOR PRINTING FACSIMILE TEST CHART 4CP01
L'OMBRE DE LA CATHEDRALE
IN DE SCHADUW VAN DE KATHEDRAAL
ISO 400
ISO 400

Figure 4. Grayscale and color test images. Color images are typeset as grayscale.

image at the encoder, therefore it uses between 60 and 800% more memory than the Kakadu encoder. However, considering the near-symmetric coding structure of a JPEG encoder and decoder, the remarkably lower memory usage of the decoder probably reveals a more accurate minimum memory requirement for JPEG. With such significant advantages in speed and memory consumption, it is possible that JPEG will remain an attractive coding method for many mobile applications.

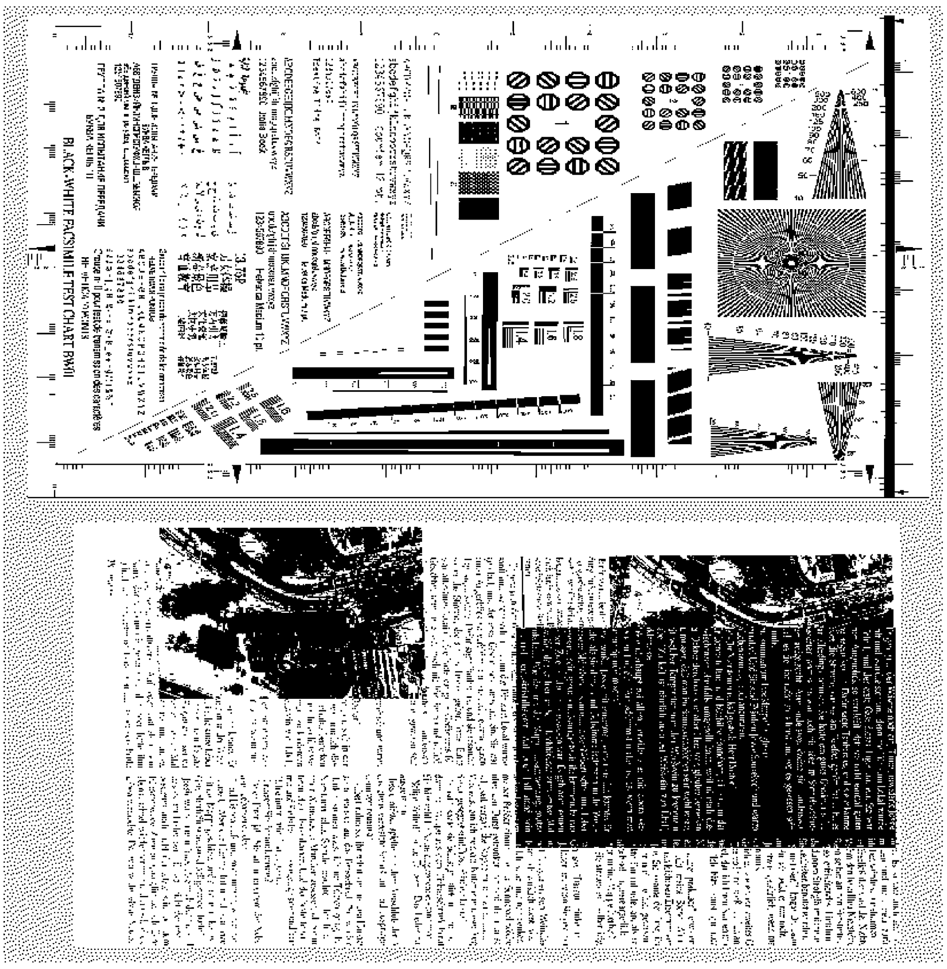


Figure 5. Bilevel test images.

The underlining wavelet coding structure of MPEG-4 VTC is very similar to JPEG 2000, therefore its performance is quite comparable with the three JPEG 2000 implementations. Although the H.264-intra coding algorithm resembles more the JPEG block coding structure, its spatial prediction feature helps it achieve a performance similar to JPEG 2000. It is interesting to see that H.264-intra performs especially well on images ULTRASOUND and CHART. This is because these images have many artificial homogeneous flat regions as well as regions with similar structures, which are well-suited for spatial prediction.

Lossless compression. Although the primary application of JPEG 2000 is lossy compression, it also supports lossless compression by using the reversible 5/3 filter bank with integer arithmetic, and bypassing quantization and the PCRD procedure. The resulting bit

Table 1. Complexity comparison of various codecs for images LENA, GOLD HILL and ULTRASOUND.

	Time (s)		Mem. (Mbytes)		Time (s)		Mem. (Mbytes)	
Codec	enc.	dec.	enc.	dec.	enc.	dec.	enc.	dec.
	LENA at 0.125 bpp				LENA at 1.00 bpp			
Kakadu	0.21	0.11	0.30	0.28	0.33	0.16	0.34	0.30
JasPer	0.86	0.50	3.19	2.41	0.91	0.59	3.24	2.41
JJ2000	2.44	1.73	2.52	2.74	2.54	1.93	2.68	3.58
MPEG-4 VTC	0.94	1.13	7.26	8.27	1.22	1.49	7.26	8.27
H.264-intra	6.86	1.27	23.63	5.96	7.90	1.40	23.63	5.96
JPEG	0.08	0.05	0.54	0.04	0.10	0.06	0.54	0.04
	GOLD HILL at 0.125 bpp				GOLD HILL at 1.00 bpp			
Kakadu	0.21	0.12	0.30	0.28	0.33	0.19	0.34	0.30
JasPer	0.90	0.50	3.22	2.41	0.98	0.59	3.28	2.41
JJ2000	2.47	1.69	2.55	2.75	2.49	1.85	2.71	3.71
MPEG-4 VTC	0.93	1.13	7.26	8.27	1.21	1.48	7.26	8.27
H.264-intra	6.90	0.61	23.63	5.96	8.03	0.74	23.63	5.96
JPEG	0.07	0.03	0.54	0.04	0.09	0.05	0.54	0.04
	ULTRASOUND at 0.125 bpp				ULTRASOUND at 1.00 bpp			
Kakadu	0.19	0.12	0.30	0.28	0.28	0.18	0.33	0.30
JasPer	0.76	0.45	2.89	2.13	0.80	0.52	2.95	2.13
JJ2000	2.34	1.69	2.41	2.76	2.33	1.84	2.53	3.34
MPEG-4 VTC	0.87	1.03	6.35	7.24	1.05	1.32	6.35	7.24
H.264-intra	6.02	0.54	20.80	5.24	6.79	0.64	20.80	5.24
JPEG	0.06	0.03	0.48	0.04	0.08	0.04	0.48	0.04

stream structure is different from lossy compression. Therefore lossless JPEG 2000 is not equivalent to lossy JPEG 2000 at a very high bit rate. Table 3 provides lossless coding performance for the codecs supporting lossless coding. JPEG-LS is specially designed for lossless coding. Its performance is relatively good with small images. However, since it does not have the capability to exploit spatial redundancy over large areas, it becomes less efficient compared to JPEG 2000 for large images. Nevertheless, its clear advantage in speed and memory usage demonstrates its viability for this particular application.

Bi-level compression. Bi-level image compression is also supported by JPEG 2000, but no special accommodations are made for this type of images. The only recommendation is to bypass the wavelet transform and apply the MQ coder with JPEG 2000 context models directly to the binary pixels. Table 3 provides performance comparisons for lossless coding of bi-level images, in which codecs were configured for bi-level input with lossless mode and without wavelet transform.

Without wavelet transform and quantization, JPEG 2000 essentially becomes a binary lossless coder, similar to JBIG and JBIG2. The major difference is in the probability models

Table 2. Complexity comparison of various codecs for images CHART and WOMAN.

	Time (s)		Mem. (Mbytes)		Time (s)		Mem. (Mbytes)	
Codec	enc.	dec.	enc.	dec.	enc.	dec.	enc.	dec.
	CHART at 0.125 bpp				CHART at 1.00 bpp			
Kakadu	2.35	1.45	0.93	0.82	4.26	2.60	1.51	1.28
JasPer	14.79	9.80	47.35	35.84	15.42	11.26	47.79	35.90
JJ2000	15.29	8.48	21.53	18.46	15.37	11.59	23.66	19.87
MPEG-4 VTC	15.79	19.12	112.96	128.83	19.47	23.83	112.96	128.83
H.264-intra	124.62	n/a	344.49	n/a	139.65	n/a	344.49	n/a
JPEG	0.94	0.33	7.62	0.05	1.12	0.52	7.62	0.05
	WOMAN at 0.125 bpp				WOMAN at 1.00 bpp			
Kakadu	8.43	5.55	2.89	2.70	11.66	7.02	3.68	3.31
JasPer	79.24	62.09	180.49	141.59	80.51	63.79	181.36	141.68
JJ2000	69.40	42.20	80.30	108.61	70.03	47.82	85.77	118.75
JPEG	2.22	1.10	15.11	0.11	2.51	1.48	15.11	0.11

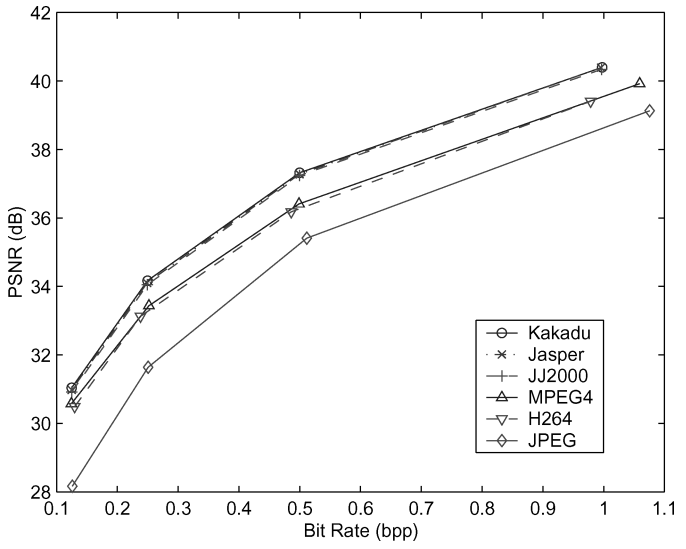
used for the entropy coder. Context classes defined by JPEG 2000 are mostly optimized for gray scale images, and are not effective for bi-level images. Therefore JPEG 2000 is outperformed by JBIG and JBIG2. JBIG and JBIG2 have clear advantages over JPEG 2000 with bi-level images because of their special designs. JBIG2 is more sophisticated than JBIG, and consumes more time and memory. However, the performance improvement is also significant. These results show that JPEG 2000 will not replace JBIG and JBIG2 in document and facsimile coding.

4.3. Progressive coding

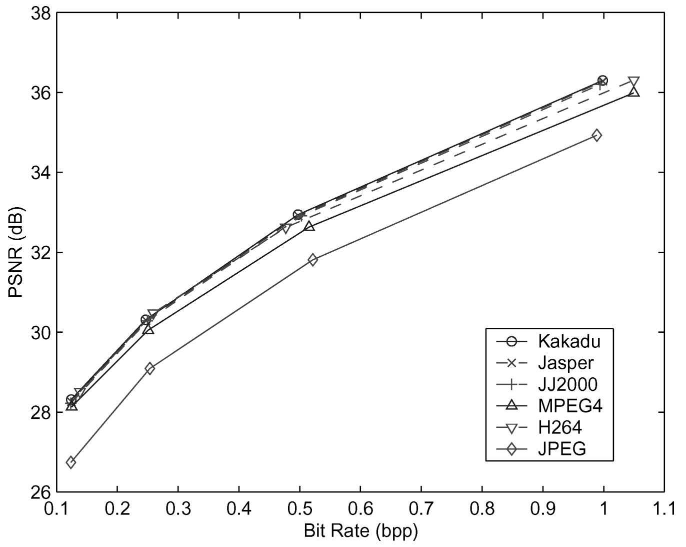
We compared JPEG 2000 and JPEG in terms of their progressive coding performance for the image BIKE. In figure 7(a), "jpeg optimal" represents the normal JPEG performance when bit streams are fully decoded at their various encoded bit rates (from 0.125 to 1.0 bpp); "jpeg progressive" represents the performance when a bit stream encoded in progressive at 1.0 bpp is truncated and decoded at various bit rates (from 0.125 to 1.0 bpp); "jpeg non-progressive" represents the same truncated decoding performance when the bit stream is encoded in non-progressive mode at 1.0 bpp. Figures 7(b) and (c) show the decoding performance of Kakadu at bit rates from 0.125 to 1.0 bpp when the bit stream is encoded at 1.0 bpp with two progression modes, LRCP and RLCP, and four different layer settings:

- 1-layer: rate/distortion is optimized at 1.0 bpp.
- 2-layer: rate/distortion is optimized at 1.0 and 0.5 bpp.
- 4-layer: rate/distortion is optimized at 1.0, 0.75, 0.5 and 0.25 bpp.
- 8-layer: rate/distortion is optimized at 1.0, 0.875, 0.75, 0.625, 0.5, 0.375, 0.25 and 0.125 bpp.

JPEG 2000 introduces the bit stream layer structure to achieve accuracy progressive encoding. It is only effective when the layer loop is the outermost loop in the progression order, i.e. LRCP progression is used. In such setting, bit stream truncation will preserve low quality layers and discard higher quality layers following the truncation point. Figure 7(b)



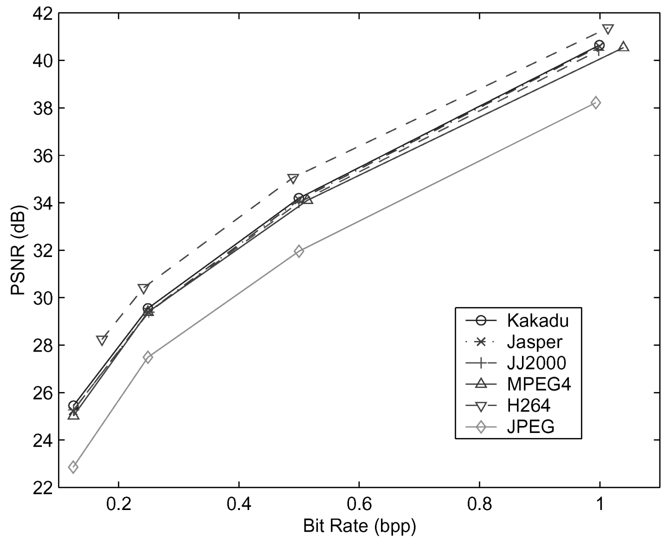
(a)



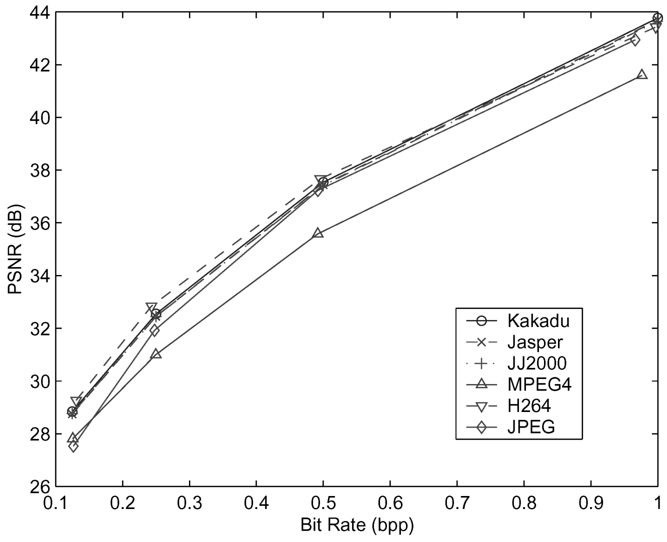
(b)

Figure 6. Performance comparison of various codecs with images: (a) LENA, (b) GOLD HILL, (c) ULTRASOUND, (d) CHART, (e) WOMAN.

(*Continued on next page*.)



(c)

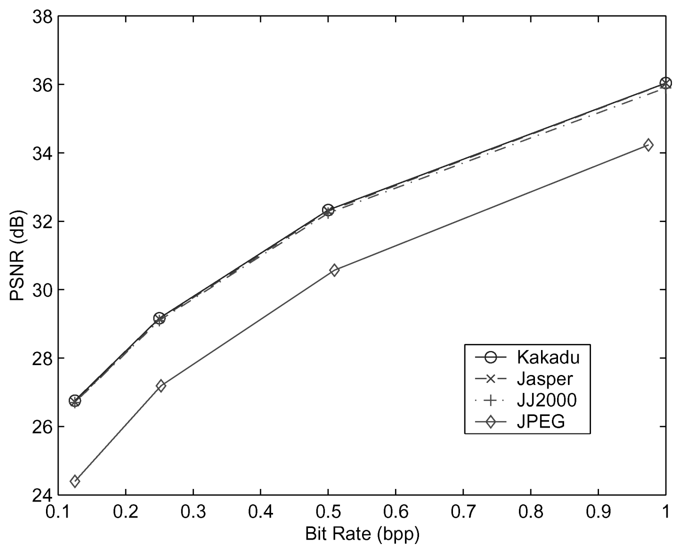


(d)

Figure 6. (*Continued*).

also reveals that layer settings at lower bit rates (e.g. 0.125 bpp, 0.25 bpp) are more effective than at higher bit rates.

RLCP represents resolution progressive coding, and its bit stream is organized by resolutions instead of layers. Figure 7(c) shows that quality optimization for lower bit rates (e.g. 0.5 bpp) is lost. However, RLCP can be used to reconstruct images at reduced sizes (by factors of 2^n).



(e)

Figure 6. (*Continued*).

Default JPEG coding is spatially progressive. A truncated bit stream will produce blank regions in the reconstructed image, yielding poor perceptual and PSNR performance. The JPEG progressive mode significantly improves its performance in such situation. The PSNR value increases smoothly at successively higher truncated bit rates. Therefore, the progressive feature of JPEG is comparable to that of JPEG 2000.

The progressive settings do not significantly affect the coding complexity and performance of Kakadu. Minor decreases in PSNR can be observed when number of layers increases. However, the progressive mode of JPEG almost doubles the computing time, and significantly increases its decoding memory usage, since the decoder will now store the whole image in its memory.

4.4. Large images

The introduction of the tile structure in JPEG 2000 makes possible the processing of very large images on low-end computing devices. This approach reduces the memory usage and computational complexity at the expense of some compression efficiency, because cross-tile correlations cannot be exploited. Another mechanism that reduces coding complexity is the arithmetic coding bypass mode, which avoids arithmetic coding at certain refinement layers. Table 4 shows the effects of tile and bypass settings on the coding performance of Kakadu, JasPer and JPEG.

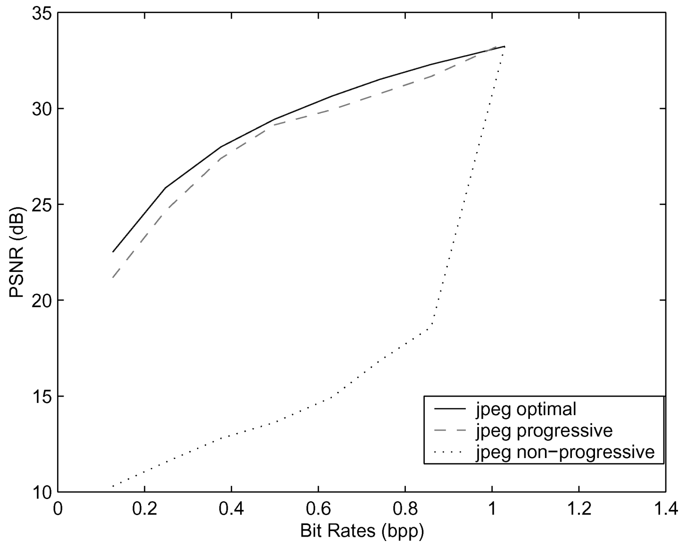
Table 3. Performance comparison of various lossless codecs for images LENA, WOMAN, CCIT1, and CCIT2.

Coder	Size (bytes)	Comp. rate	Enc. time (s)	Dec. time (s)	Enc. mem. (Mbytes)	Dec. mem. (Mbytes)
			LENA lossless			
Kakadu	141089	1.8580	0.38	0.40	0.43	0.40
JasPer	141116	1.8576	0.70	0.57	3.52	2.42
JJ2000	142148	1.8442	2.63	2.10	3.23	5.62
JPEG	153839	1.7040	0.28	0.10	3.01	0.03
JPEG-LS	138809	1.8885	0.14	0.14	0.03	0.03
			WOMAN lossless			
Kakadu	7537793	2.0866	18.94	19.48	11.02	10.35
JasPer	7538026	2.0866	59.00	50.75	193.79	141.96
JJ2000	7572340	2.0771	60.47	n/a	117.60	n/a
JPEG	10080849	1.5602	11.91	5.84	60.04	0.07
JPEG-LS	8991690	1.7492	8.76	8.31	0.05	0.06
			CCIT1 lossless			
Kakadu	150495	13.8244	2.54	1.91	0.87	0.85
JasPer	151533	13.7297	13.58	11.76	187.53	150.04
JJ2000	157960	13.1711	11.57	n/a	99.59	n/a
JBIG2	83505	24.9147	14.35	13.49	4.81	4.32
JBIG	93868	22.1641	2.04	1.68	2.51	2.51
			CCIT2 lossless			
Kakadu	543219	3.0764	3.94	3.38	1.19	1.19
JasPer	544287	3.0704	13.18	11.40	151.01	120.57
JJ2000	549607	3.0407	14.00	n/a	82.26	n/a
JBIG2	168640	9.9097	11.95	10.70	4.11	3.62
JBIG	214192	7.8022	1.51	1.45	2.03	2.02

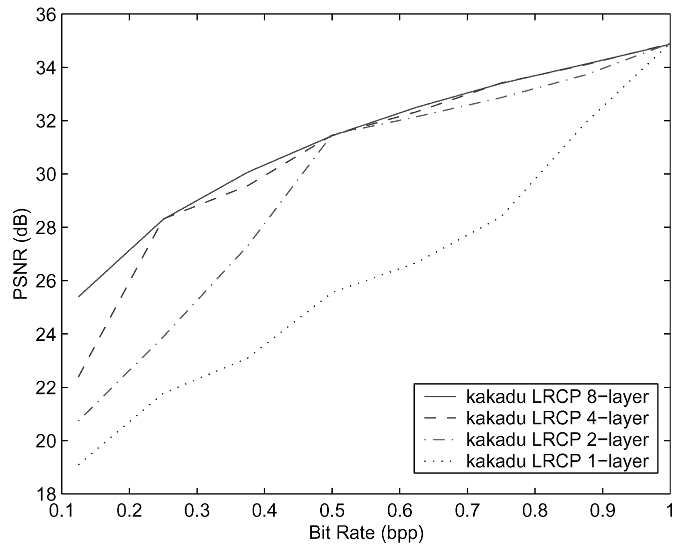
The advantage of using multiple tiles is clearly demonstrated by the results for the JasPer implementation. The 64×64-pixel tile setting reduces its memory usage by a factor of ten, and almost doubles its processing speed. The cost is a notable loss in PSNR performance. On the other hand, Kakadu complexity does not seem to be significantly affected by different tile settings. This is explained by its efficient implementation using pipelining, as described in Section 2.5. As expected, the PSNR loss is more pronounced for WOMAN, a relatively smooth image with strong global correlations, than for CAFE, a highly detailed image with different regional structures and therefore less cross-tile correlation. The bypass mode does not have a noticeable impact on any performance measure. It has been found that the bypass mode in fact only generates a small portion of the output bit stream.

4.5. Error resilience

JPEG 2000 defines several error resilient mechanisms working at both the entropy coding level and at the bit stream packet level. These include

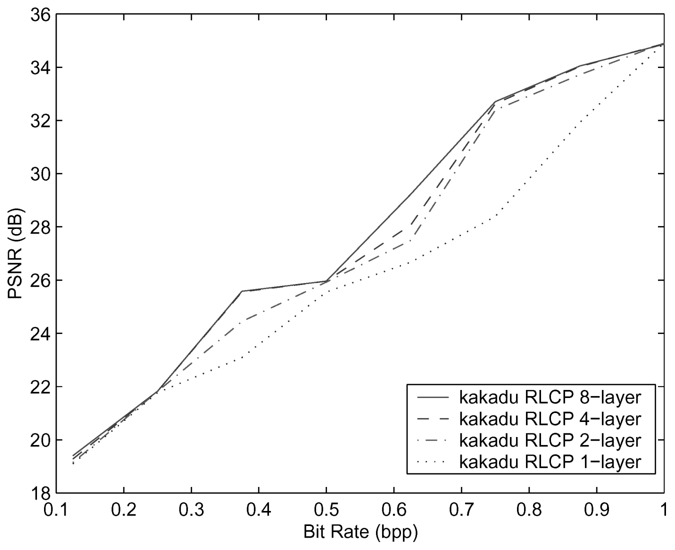


(a)



(b)

Figure 7. Progressive coding performance results for (a) JPEG, (b) JPEG 2000 with LRCP progression, and (c) JPEG 2000 with RLCP progression. (*Continued on next page*.)



(c)

Figure 7. (*Continued*).

- terminating and restart arithmetic coding at each pass with predictable rules (PRED-TERM/RESTART) that can synchronize the decoder with the encoder;
- inserting special arithmetic coded segmentation marker at the end of each bit-plane (SEGMARK);
- resetting the arithmetic coding probability context at each new pass (RESET-PROB);
- using the selective arithmetic coding bypass mode (BYPASS);
- inserting a re-synchronization marker (SOP) in the header of each packet.

The error handling method suggested by JPEG 2000 requires the capability to detect and isolate an error event. Once an error is located at a certain layer within a block (subband or tile), the following layers of this block (subband or tile) will not be decoded. This can prevent a corrupted bit stream from degrading the previously decoded correct information. Kakadu is used in the test because it is the only implementation supporting error tolerance. The decoder option “-resilient” is enabled.

These methods and some of their combinations have been tested in a simulated random noise channel with bit error rate (BER) ranging from 10^{-5} to 10^{-3}. Each recorded entry represents 100 trials. Results shown in Tables 6 and 7 include the number of decodeable trials, and the mean, maximum, minimum and standard deviation of PSNR values from all decoded bit streams. Decodeability depends mostly on the implementation, and less on channel condition. An implementation can terminate the decoding process upon a minor bit stream error. Therefore it is not an error resilient measure. Instead it represents a relative sense of error severeness and an adjustment factor for the rest of the PSNR values.

Table 4. Performance of codecs for large images CAFE and WOMAN at 1.0 bpp.

Coder setting	PSNR (dB)	Enc. time (s)	Dec. time (s)	Enc. mem. (Mbytes)	Dec. mem. (Mbytes)
		CAFE			
Kakadu, 1-tile	32.07	5.51	3.36	1.86	1.59
Kakadu, 4-tile	32.06	5.56	3.37	1.86	1.23
Kakadu, 16-tile	32.03	5.69	3.44	2.03	1.18
Kakadu, 64-tile	31.93	6.21	3.66	2.62	1.32
Kakadu, BYPASS	32.00	4.86	3.21	1.86	1.59
JasPer, 1-tile	32.04	31.54	21.90	63.73	47.33
JasPer, 4-tile	31.12	27.45	18.63	19.96	15.63
JasPer, 16-tile	30.25	19.79	10.88	8.93	7.71
JasPer, 64-tile	29.61	19.55	10.03	6.25	5.76
JasPer, BYPASS	31.98	30.09	21.96	63.75	47.33
JPEG, 1-tile	30.18	1.57	0.79	10.05	0.05
		WOMAN			
Kakadu, 1-tile	38.45	5.70	3.31	1.88	1.59
Kakadu, 4-tile	38.44	5.81	3.32	1.91	1.33
Kakadu, 16-tile	38.41	5.94	3.40	2.07	1.20
Kakadu, 64-tile	38.18	6.44	3.57	2.66	1.33
Kakadu, BYPASS	38.40	4.90	3.08	1.87	1.59
JasPer, 1-tile	38.44	29.26	21.94	62.53	47.31
JasPer, 4-tile	37.26	25.45	18.64	19.69	15.63
JasPer, 16-tile	35.29	17.75	10.80	8.91	7.71
JasPer, 64-tile	34.09	17.45	9.91	6.24	5.74
JasPer, BYPASS	38.40	28.52	21.89	62.55	47.31
JPEG, 1-tile	36.66	1.56	0.77	10.05	0.05

Test results show that PRED-TERM/RESTART and SEGMARK are the most effective methods for detecting errors. In fact these two methods work similarly. The correct decoding of a segmentation marker or correct termination of a bit plane coding pass indicates the correctness of the bit stream up to the current point. The combination of PRED-TERM/RESTART and SEGMARK does not further improve performance. This is expected because the mechanisms are similar, and there is little chance that an error can pass one of them but be detected by the other.

If the arithmetic coding is not terminated, the SOP in the next packet header may be overrun if error occurs, and therefore it is not an effective error detection method by its own. When SOP is combined with PRED-TERM/RESTART, the performance is only slightly improved.

Table 5. Time profiling (in%) of major encoder (a) and decoder (b) components, for the color image WOMAN.

Codec	Bit rate (bpp)	Pixel conv.	Color trans.	Wavelet trans.	Coeff. conv.	Sig. pass	Ref. pass	Cleanup pass
				(a)				
Kakadu	0.25	2.6	7.3	46.7	13.9	9.6	4.1	12.3
Kakadu	1.0	2.4	6.2	36.8	10.6	18.6	6.5	14.3
JasPer	0.25	5.1	2.7	66.1	2.8	4.8	2.7	8.1
JasPer	1.0	4.8	2.5	64.8	2.6	4.9	2.6	8.1

Codec	Bit rate (bpp)	Sig. pass	Ref. pass	Cleanup pass	Coeff. conv.	Wavelet trans.	Color trans.	Pixel conv.
				(b)				
Kakadu	0.25	2.8	0.9	3.3	3.1	74.8	8.7	4.8
Kakadu	1.0	9.8	4.3	7.5	3.8	57.1	9.3	5.8
JasPer	0.25	0.3	0.2	0.3	2.5	79.1	2.1	5.3
JasPer	1.0	1.3	0.5	1.4	2.5	78.5	1.8	5.5

The purpose of the RESET-CONTEXT and BYPASS methods is to limit error propagation, not to detect errors. RESET-CONTEXT only helps slightly when combined with PRED-TERM/RESTART, and BYPASS does not have clear advantage by itself or combined with PRED-TERM/RESTART.

The noisy channel performance of the JPEG 2000 codec is mostly determined by the actual number of bit errors and the position of the first bit error in the bit stream. Because the image GOLD HILL has a much shorter bit stream, it will have much fewer bit errors than the CAFE bit stream at the same BER. Therefore, the error effects in GOLD HILL are less severe than those in CAFE. This situation also explains why a bit stream encoded at a higher bit rate may have a lower PSNR in a noisy environment.

All these error resilience tools are generic methods used to combat random errors. Given the JPEG 2000 error handling approach, bursty errors can be translated into random errors since any number of errors occurring in one coding pass are equivalent to a single error in the same pass. Therefore, having many consecutive errors is preferred to to the worst-case scenario featuring random isolated errors.

4.6. Summary of experimental results

Photographic images: For natural images with consistent global structures (e.g. LENA and WOMAN), JPEG 2000 clearly outperforms all other codecs. This is where the DWT coding structure achieves its best efficiency. For natural images with uncorrelated regional structures or detailed textures (e.g. GOLD HILL, BIKE, and CAFE), JPEG 2000's advantage over other codecs becomes less significant.

Table 6. Noisy channel simulation for the image GOLD HILL.

	at 0.5 bpp					at 0.25 bpp				
BER	Decodeable (%)	Mean (dB)	Max (dB)	Min (dB)	Std.	Decodeable (%)	Mean (dB)	Max (dB)	Min (dB)	Std
					No error resilience feature					
0.00001	100	32.00	32.94	25.58	1.47	100	29.80	30.31	23.88	1.12
0.0001	100	24.62	31.05	10.09	4.47	100	26.23	30.31	9.36	3.10
0.001	98	14.25	20.70	6.09	3.84	97	14.77	21.93	6.09	4.74
					With PRED-TERM/RESTART					
0.00001	100	31.87	32.82	22.71	2.06	100	29.80	30.24	23.99	1.05
0.0001	100	26.57	31.66	14.25	3.76	100	26.59	30.24	14.25	3.04
0.001	98	19.73	22.76	14.25	2.85	98	20.00	23.49	14.25	3.03
					With SEGMARK					
0.00001	100	32.32	32.94	26.45	1.05	100	29.96	30.27	25.67	0.65
0.0001	100	27.76	31.41	14.25	3.01	100	27.32	30.27	14.25	2.59
0.001	98	20.13	24.48	13.65	3.33	98	19.79	25.11	8.08	3.41
					With SOP marker					
0.00001	100	31.81	32.94	18.30	2.13	100	29.73	30.31	17.87	1.51
0.0001	100	24.77	30.87	11.07	4.10	100	25.57	30.31	10.24	3.81
0.001	98	15.02	21.29	6.71	3.34	98	16.26	22.15	7.40	3.39
					With BYPASS					
0.00001	100	31.43	32.94	6.31	3.29	100	29.63	30.30	7.42	2.43
0.0001	100	24.62	30.63	6.30	4.61	100	25.68	30.30	7.42	4.36
0.001	98	13.38	20.24	6.24	3.96	98	14.44	23.60	6.23	4.65

Synthetic and medical images: For computer graphics, compound images and some medical images (e.g. CHART and ULTRASOUND), the block-based H.264-intra codec appears to be more efficient than JPEG 2000 because of its more efficient coding of inter-block correlation within a single frame.

Bi-level images: JBIG and JBIG2 perform significantly better than JPEG 2000.

Lossless coding: JPEG 2000 in lossless mode and JPEG-LS achieve similar performance on lossless coding, however JPEG-LS is much faster.

Large images: With traditional implementations (e.g. JasPer), tile partition is an effective tool to balance memory usage with coding performance. It can also facilitate efficient parallel processing, especially for images containing several less correlated regions.

Progressive coding: LRCP progression provides fine scale SNR progressive coding, while RLCP progression provides resolution progressive coding.

Error resilience: Both PRED-TERM/RESTART and SEGMARK are effective error resilient mechanisms. However, combinations of these or other settings will not achieve noticeable further protection.

Table 7. Noisy channel simulation for the image CAFE.

	at 0.5 bpp					at 0.25 bpp				
BER	Decodeable (%)	Mean (dB)	Max (dB)	Min (dB)	Std.	Decodeable (%)	Mean (dB)	Max (dB)	Min (dB)	Std
					No error resilience feature					
0.00001	96	21.68	26.08	12.48	4.26	96	20.65	23.11	13.44	2.71
0.0001	86	11.80	17.81	8.00	2.06	83	12.41	17.99	7.07	2.43
0.001	66	8.09	10.01	6.59	0.68	67	7.97	10.35	6.08	0.80
					With PRED-TERM/RESTART					
0.00001	100	22.13	26.33	13.54	3.50	100	20.67	23.00	12.62	2.59
0.0001	99	14.59	19.64	10.48	2.15	99	14.56	19.03	10.48	2.16
0.001	97	10.94	11.45	10.48	0.28	97	10.92	11.44	10.48	0.29
					With SEGMARK					
0.00001	100	22.85	26.33	13.40	3.58	100	21.20	23.08	13.13	2.54
0.0001	98	14.85	20.86	10.48	2.44	96	14.60	19.61	10.48	2.26
0.001	97	10.62	11.38	8.39	0.51	97	10.64	11.44	8.20	0.56
					With SOP marker					
0.00001	99	21.34	26.13	11.09	4.65	100	20.06	23.00	12.08	3.31
0.0001	84	11.54	18.33	8.37	2.32	96	12.17	17.71	7.36	2.29
0.001	73	7.57	9.39	5.71	0.68	78	7.35	10.33	5.55	0.71
					With BYPASS					
0.00001	100	21.15	25.95	10.92	4.84	100	19.91	23.02	10.40	3.57
0.0001	99	11.84	18.22	8.46	2.04	98	12.11	17.75	8.24	2.32
0.001	97	8.10	10.64	5.90	0.81	96	8.10	10.48	6.39	0.76

4.7. Topics for future study

JPEG 2000 has inferior performance for synthetic images as well as bi-level images. The H.264-intra coding algorithm benefits from its flexible spatial prediction of an encoded block. Significant portions of synthetic and bi-level images (such as flat areas or line drawings) can be efficiently predicted from previously coded adjacent blocks, resulting in a very small number of bits. Based on the same idea, improvements can be made to the context modeling in JPEG 2000 binary arithmetic coding. In the current standard specification, contexts are defined within a region surrounding the encoded DWT domain sample. More sophisticate context models can be designed to include arbitrary encoded samples as long as this context formation can be explicitly or implicitly encoded in the bitstream. This extension could exploit correlations between repeated patterns in synthetic and bi-level images.

When coding large images, it is usually desirable to partition them into tiles if the coding performance is not sacrificed. Large images tend to contain many objects and scenes that are less correlated. For each such image, an optimal tile partition that can minimize the loss of coding efficiency may be determined by calculating a well-defined correlation parameter.

Progressive decoding is an important feature for many applications. However JPEG 2000 and all other DWT based coding algorithms are computatioinally progressive only in encoding. The decoder has to perform a new inverse DWT for each new accuracy or resolution level from the progressive bitstream. Because the IDWT accounts for a significant amount of decoding computation, algorithmic improvements that avoid the multiple IDWT computations would be very valuable.

Error resilience features specified in the standard are very basic, yet effective error detection tools. Because of the embedded nature of JPEG 2000 bitstream, usually the bits after the first detected bit error will not contribute to any quality improvement at the decoder. For a certain channel BER, the probability of a packet or segment containing at least one bit error can be estimated. With this prediction, it is desirable to truncate this bitstream at this point before it is sent into the channel. This scheme has clear advantages in TCP/IP networks. The channel BER is usually time varying, so the challenge is to devise an statistically optimal adaptive bitstream truncation policy.

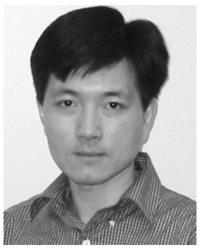
5. Conclusion

A comprehensive performance analysis of the JPEG 2000 image coding standard was presented. This study was conducted with various standard image coding tools currently available. Although it was not intended to be exhaustive, most of the JPEG 2000 coding algorithm, structure specifications, as well as frequently used features were discussed. The experimental results provide an objective view of the advantages and disadvantages of this new standard. It is evident that JPEG 2000 will be adopted in many applications that requires high image quality over bandwidth constrained channels and media. However, it is unlikely that it will become the only image coding tool to be used in still image coding applications.

References

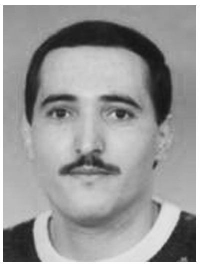
1. M.D. Adams, "The JPEG-2000 still image compression standard," ISO/IEC JTC 1/SC 29/WG 1 N 2412. Available online from `http://www.ece.uvic.ca/~mdadams` and distributed with the JasPer software, 2001.
2. M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," IEEE Transactions on Image Processing, Vol. 2, No. 1, pp. 205–220, 1992.
3. C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG 2000 still image coding system: An overview," IEEE Transactions on Consumer Electronics, Vol. 46, pp. 1103–1127, 2000.
4. ISO/IEC MPEG and ITU-T VCEG Joint Video Team (JVT), "H.26L Reference Software, version 8.4," ftp://ftp.imtc-files.org/jvt-experts, 2002.
5. JJ2000: An Implementation of the JPEG 2000 standard in Java, version 4.1: `http://jj2000.epfl.ch`, 2000.
6. JTC1/SC29/WG1: "11544: Information technology—Coded representation of picture and audio information—Progressive bi-level image compression," International standard, ISO/IEC, 1993.
7. JTC1/SC29/WG1: "10918-1: Information technology—Digital compression and coding of continuous-tone still images: Requirements and guidelines," International standard, ISO/IEC, 1994.
8. JTC1/SC29/WG1: "Call for contributions for JPEG 2000 (JTC 1.29.14, 15444): Image Coding System," Approved call for contributions, ISO/IEC, 1997.

9. JTC1/SC29/WG1: "14495-1: Information technology—Lossless and near-lossless compression of continuous-tone still images: Baseline," International Standard, ISO/IEC, 2000a.
10. JTC1/SC29/WG1: "15444-1: Information technology—JPEG 2000 image coding system—Part 1: Core coding system," International Standard, ISO/IEC, 2000b.
11. JTC1/SC29/WG1: "14492: Information technology—Lossy/lossless coding of bi-level images," International Standard, ISO/IEC, 2001.
12. JTC1/SC29/WG11: "14496-2: Information technology—Coding of audio-visual objects—Part 2: Visual," International Standard, ISO/IEC, 2001.
13. JTC1/SC29/WG11: "14496-10: Information technology—Coding of audio-visual objects—Part 10," Draft international Standard, ISO/IEC, 2002.
14. M.W. Marcellin, M.J. Gormish, A. Bilgin, and M.P. Boliek, "An overview of JPEG-2000," in Data Compression Conference. Snowbird, UT, 2000, pp. 523–544.
15. Markus Kuhn, "JBIG-kit, version 1.2," 2000, `http://www.cl.cam.ac.uk/~mgk25/jbigkit`.
16. Microsoft Corporation, 2000, "Microsoft MPEG-4 Visual Reference Software, version 2.3.0,"
17. M. Rabbani and R. Joshi, "An overview of the JPEG2000 still image compression Standard," Signal Processing: Image Communication, Vol. 17, No. 1, pp. 3–48, 2002.
18. D. Santa-Cruz and T. Ebrahimi, "An analytical study of JPEG 2000 functionalities," in International Conference on Image Processing, Vancouver, 2000, BC, pp. 49–52.
19. J.M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," IEEE Transactions on Signal Processing, Vol. 41, No. 12, pp. 3445–3462, 1993.
20. D. Taubman, "High performance scalable image compression with EBCOT," IEEE Transactions on Image Processing, Vol. 9, pp. 1158–1170, 2000.
21. D. Taubman, "Kakadu: A comprehensive, heavily optimized, fully compliant software toolkit for JPEG2000 developers, version 3.2," 2002, `http://maestro.ee.unsw.edu.au/~taubman/kakadusoftware`.
22. D. Taubman and M. Marcellin, Jpeg2000: Image Compression Fundamentals, Standards, and Practice, Kluwer Academic Publishers, 2001.
23. The Independent JPEG Group, "JPEG Image Compression Library, version 6b," 1998, `http://www.ijg.org/files`.
24. The JasPer project, version 1.500.4. 2000, `http://www.ece.uvic.ca/ mdadams/jasper`.
25. UBC Signal Processing and Multimedia Group, "JPEG-LS Public Domain Code, version 2.2," `http://www.ece.ubc.ca/spmg/research/jpeg/jpeg_ls/jpegls.html`, 1999.
26. UBC Signal Processing and Multimedia Group, "JBIG2 Software, version 1.0.0," 2000, `http://spmg.ece.ubc.ca/ jbig2`.
27. M. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," IEEE Transactions on Image Processing, Vol. 9, pp. 1309–1324, 2000.
28. T. Wiegand, "Working Draft Number 2, Revision 2 (WD-2), JVT-B118r2," Technical report, ISO/IEC/ITU-T, 2002.
29. T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G.J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," to appear in the IEEE Transactions on Circuits and Systems for Video Technology, 2002.



Hong Man received his Ph.D. degree from Georgia Institute of Technology in 1999, in Electrical Engineering. He joined Stevens Institute of Technology in 2000, and currently he is an assistant professor in the Department of Electrical and Computer Engineering. He is serving as the director for Visual Information Environment Laboratory at Stevens, the director for Computer Engineering undergraduate program in the ECE department, and the coordinator for NSA Center of Academic Excellence in Information Assurance in the School of Engineering. He is a member of the IEEE and ACM. He served as member of organizing committee for IEEE International Workshop on Multimedia and Signal Processing (MMSP) 2002 and 2005, member of technical program committee for IEEE Vehicular Technology Conference (VTC) Fall 2003, and IEEE/ACM International Conference on E-Business and Telecommunication Networks (ICETE) 2004 and 2005. He is a committee member on IEEE SPS TC for Education. He was an active contributor to the ISO/ITU JPEG 2000 image coding standard.

Alen Docef received his Diploma of Engineer from the Polytechnic Institute of Bucharest, Romania, in 1991. He obtained an M.S.E.E degree in 1992 and a Ph.D. degree in 1998 from the Georgia Institute of Technology, Atlanta, Georgia, all in electrical engineering. From 1998 to 1999 he worked as a research engineer in the Signal Processing and Multimedia Group of the University of British Columbia. In 2000 he joined the Virginia Commonwealth University School of Engineering as an Assistant Professor. His research interests include multimedia signal compression, medical image processing, and real-time implementation of DSP algorithms. He has been a member of the IEEE since 1995.



Faouzi Kossentini received the B.S., M.S., and Ph.D. degrees from the Georgia Institute of Technology, Atlanta, in 1989, 1990, and 1994, respectively. He is presently the President and CEO of UB Video Inc., a company in Vancouver (Canada) that develops video communication products for the video conferencing and broadcast markets. Before the year 2004, he had been an associate professor in the Department of Electrical and Computer Engineering at the University of British Columbia, where he was involved in research in the areas of signal processing, communications and multimedia. He has co-authored more than two hundred journal papers, conference papers and book chapters. Dr. Kossentini is a senior member of the IEEE. He has served as a Vice General Chair for ICIP-2000, and he has also served as an associate editor for the IEEE transactions on Image Processing and the IEEE transactions on Multimedia.