On the JPEG2000 Implementation on Different Computer Platforms

E. B. Christopoulou[a,∴], A. N. Skodras[a,b], T. R. Reed[c] and C. A. Christopoulos[d]

[a]*Electronics Laboratory, University of Patras*
Patras GR-26110, Greece
[b]*Computer Technology Institute, University of Patras*
Patras GR-26110, Greece
[c]*Electrical and Computer Engineering, University of California*
Davis CA 95616, USA
[d]*Media Lab, Ericsson Research, Ericsson Radio Systems AB*
Stockholm S-16480, Sweden

ABSTRACT

JPEG2000 is the new standard for the compression of still images. Its objective and subjective image quality performance is superior to existing standards. In this paper the Part I of the JPEG2000 standard is presented in brief, and its implementation complexity on different computer platforms is reported.

Keywords: JPEG, wavelet transform, image compression

1. INTRODUCTION

With the continual expansion of multimedia and Internet applications, the needs and requirements of the technologies used grew and evolved. In March 1997 a call for contributions[1] was launched for the development of a new standard for the compression of still images, JPEG2000. This project, JTC* 1.29.14 (15444), was intended to create a new image coding system for different types of still images (bi-level, grey-level, colour, multi-component), with different characteristics (natural images, scientific, medical, remote sensing, text, rendered graphics, etc.) allowing different imaging models (client/server, real-time transmission, image library archival, limited buffer and bandwidth resources, etc.) preferably within a unified system. This coding system should provide low bit-rate operation with rate-distortion and subjective image quality performance superior to existing standards, without sacrificing performance at other points in the rate-distortion spectrum, incorporating at the same time many contemporary features.

The standardization process, which is coordinated by the JTC1/SC29/WG1 of ISO/IEC# has already (as of May 2000) produced the Final Committee Draft Document of Part I of the standard[2]. The Draft International Standard (DIS) is scheduled for August 2000 and the International Standard (IS) for December 2000. Only editorial changes are expected at this stage and therefore, from a technical point of view, Part I of the standard is 'frozen'.

In the present paper the structure of Part I of the JPEG2000 standard is presented and implementation results on different computer platforms are reported. The paper is organized in the following way: In Section 2 the main features of the new standard are presented. The architecture of the standard is described in Section 3, including tiling, multi-component

∴Corresponding author. Other author information: E.B.C.; Email: echris@physics.upatras.gr; A.N.S.; Email: skodras@cti.gr; T.R.R.; Email: trreed@ucdavis.edu; C.A.C.; Email: Charilaos.Christopoulos@era.ericsson.se

* JTC stands for Joint Technical Committee

SC, WG, IEC stand for Standing Committee, Working Group and International Electrotechnical Commission, respectively.

transformations, wavelet transforms, quantisation and entropy coding. The implementation results are reported in Section 4 of the paper, while Section 5 concludes the paper.

2. PRIMARY FEATURES OF THE STANDARD

The JPEG2000 standard provides a set of features that are of importance to many high-end and emerging applications, by taking advantage of new technologies. It addresses areas where current standards fail to produce the best quality or performance and provides capabilities to markets that currently do not use compression. The markets and applications better served by the JPEG2000 standard are Internet, colour facsimile, printing, scanning (consumer and pre-press), digital photography, remote sensing, mobile telephony, medical imagery, digital libraries / archives and E-commerce. Each application area imposes some requirements that the standard should fulfil. Some of the most important features that this standard possesses are the following:[3]

Superb low bit-rate performance: This standard offers performance superior to the current standards at low bit-rates (e.g. below 0.25 bpp for highly detailed grey-scale images). This significantly improved low bit-rate performance is achieved without sacrificing performance over the rest of the rate-distortion spectrum. Examples of applications that need this feature include network image transmission and remote sensing. This is the highest priority feature.

Lossless and lossy compression: The standard provides lossless compression naturally in the course of progressive decoding. Examples of applications that can use this feature include medical images, where loss is not always tolerated, image archival applications, where the highest quality is vital for preservation but not necessary for display, network applications that supply devices with different capabilities and resources, and pre-press imagery. The standard also creates embedded bitstreams, and allows progressive lossy to lossless applications.

Progressive transmission by pixel accuracy and resolution: Progressive transmission that allows images to be reconstructed with increasing pixel accuracy or spatial resolution is essential for many applications. This feature allows the reconstruction of images with different resolutions and pixel accuracy, as needed or desired for different target devices. Examples of applications include the World Wide Web, image archival applications and printers.

Random codestream access and processing: This feature allows user defined regions of interest in the image to be randomly accessed and/or decompressed with less distortion than the rest of the image. Also, random codestream processing allows operations such as rotation, translation, filtering, feature extraction and scaling.

Robustness to bit-errors: One application where this is important is transmission over wireless communication channels. Portions of the codestream may be more important than others in determining decoded image quality. Proper design of the codestream can aid subsequent error correction systems in alleviating catastrophic decoding failures.

Region-of-Interest (ROI):[4] One of the features included in JPEG2000 is ROI coding, in which certain regions of the image can be coded with better quality than the rest of the image (background). The ROI approach defined in JPEG2000 Part I is called the MAXSHIFT method, and allows ROI encoding of arbitrary shaped regions without the need of shape information and shape decoding. This is achieved by scaling the coefficients up, so that the bits associated with the ROI are placed in higher bit-planes. During the embedded coding process, those bits are placed in the bit-stream before the non-ROI parts of the picture. Thus, the ROI is decoded or refined before the rest of the picture. Regardless of the scaling, a full decoding of the bit-stream results in a reconstruction of the whole picture with the highest fidelity available. If the bit-stream is truncated, or the encoding process is terminated before the whole image is fully encoded, the ROI will have a higher fidelity than the rest of the image.

Scalability:[5] Scalable coding of still pictures means the ability to achieve coding of more than one resolution and/or quality simultaneously. Scalable image coding involves generating a coded representation (bitstream) in a manner which facilitates the derivation of images of more than one resolution and/or quality by scalable decoding. Bitstream scalability is the property of a bitstream that allows decoding of appropriate subsets of a bitstream to generate complete pictures of resolution and/or quality commensurate with the proportion of the bitstream decoded. Decoders of different complexities, from low performance to high performance can coexist when using a truly scalable bitstream. While low performance decoders may decode only small portions of the bitstream producing basic quality, high performance decoders may decode much more and produce significantly higher quality. The most important types of scalability are signal-to-noise (SNR) scalability and

spatial scalability. *SNR scalability* involves generating at least two image layers of the same spatial resolution, but different qualities, from a single image source. *Spatial scalability* is intended for use in systems which require a minimum of two layers of spatial resolution. Spatial scalability involves generating at least two spatial resolution layers from a single source such that the lower layer is coded by itself to provide the basic spatial resolution and the enhancement layer employs the spatially interpolated lower layer and carries the full spatial resolution of the source image.

Error Resilience: To improve the performance of transmitting compressed images over error prone channels, error resilient bit stream syntax and tools are included in this standard. The error resilient tools deal with channel errors using the following approaches: data partitioning and re-synchronisation, error detection and concealment, and Quality of Service (QoS) transmission based on priority.

Multicomponent Images: JPEG2000 supports multiple-component images. Different components need not have the same bit-depths; nor need they all be signed or unsigned. For reversible (lossless) systems, the only requirement is that the bit-depth of each output image component must be identical to the bit-depth of the corresponding input image component. Two different component transformations are supported by the standard: an *irreversible component transformation* (ICT) and a *reversible component transformation* (RCT).

New File Format with IPR Capabilities: An optional file format (JP2) for the JPEG2000 compressed image data has been defined by the standard. This format has provisions for both image and metadata, a mechanism to indicate the tonescale or colourspace of the image, a mechanism by which readers may recognize the existence of intellectual property rights (IPR) information in the file and a mechanism by which metadata (including vendor specific information) can be included in the file.

3. ARCHITECTURE OF THE STANDARD

The block diagram of the JPEG2000 encoder [2 6 7] is illustrated in Fig. 1a. The discrete transform is first applied to the source image data. The transform coefficients are then quantised and entropy coded, before forming the output codestream (bitstream). The decoder is depicted in Fig. 1b. The codestream is first entropy decoded, dequantised and inverse discrete transformed, thus resulting in the reconstructed image data. A discrete wavelet transform (DWT) is employed in this standard and not the discrete cosine transform. It is worth mentioning that, unlike other coding schemes, JPEG2000 can be both lossy and lossless. This depends on the wavelet transform and the quantisation applied. The standard allows tiling of the image, i.e. the partitioning of the image into rectangular non-overlapping blocks. Tiles are encoded independently. Prior to computation of the forward discrete wavelet transform on each tile, all samples of the image tile component are DC level shifted by subtracting the same quantity (i.e. the component depth) from each sample.

Arithmetic coding is used in the last part of the encoding process. The MQ coder is adopted in JPEG2000. This coder is basically similar to the QM-coder adopted in the original JPEG standard. The MQ-coder is also used in the JBIG-2 standard and is available on a royalty and fee free basis for ISO standards. The development of a royalty and fee free standard was one of the main goals of WG1, since this is important for the standard to become widely accepted, in the same manner as the original JPEG with Huffman coding is now.[8]

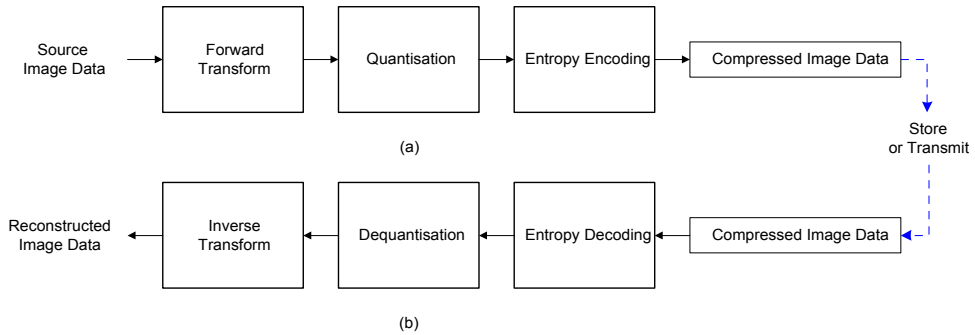

Figure 1. Block diagram of the JPEG2000 (a) encoder and (b) decoder.

The JPEG2000 encoding procedure is as follows:

- The image is decomposed into components.
- The image and its components can be decomposed into rectangular tiles. The tile-component is the basic unit of the original or reconstructed image.
- The wavelet transform is applied to each tile, decomposing it into different resolution levels.
- These decomposition levels are made up of sub-bands of coefficients that describe the frequency characteristics of local areas of the tile-component, rather than across the entire tile-component.
- Sub-bands of coefficients are quantised and partitioned into rectangular arrays of "code-blocks".
- Bit-planes of the coefficients of a "code-block" are entropy coded.
- The encoding can be done such that certain ROI's can be coded with higher quality as compared to the background.
- Markers are added to the bitstream to allow error resilience.
- The codestream has a main header at the beginning that describes the original image and the various decomposition and coding styles that have to be used to locate, extract, decode, and reconstruct the image with the desired resolution, fidelity, region of interest or other characteristics.

Let us now look at the main encoding blocks in greater detail.

3.1. Wavelet Transform

Tile components are decomposed into different decomposition levels using a discrete wavelet transform. These decomposition levels contain a number of subbands, which consist of coefficients that describe the horizontal and vertical spatial frequency characteristics of the original tile component. Power of 2 (dyadic) decompositions are allowed in Part I, as shown in Fig. 2.

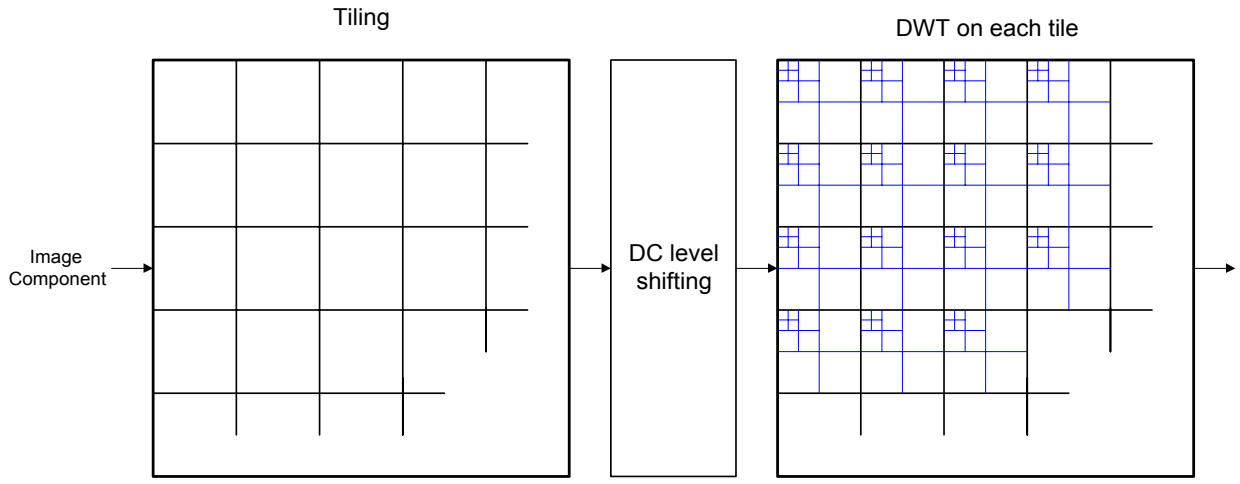

Figure 2. Tiling, DC level shifting and DWT of each image tile component.

To perform the forward DWT the standard uses a 1-D subband decomposition, applied to the rows and columns of each tile to produce low-pass and high-pass samples. Low-pass samples represent a downsampled low-resolution version of the original set, and high-pass samples represent a downsampled residual version of the original set, needed for the perfect reconstruction of the original set from the low-pass set. The low-pass set is decomposed successively. The DWT can be *irreversible* or *reversible*. The default *irreversible transform* is implemented by means of the Daubechies 9-tap/7-tap filter.[9] The analysis and the corresponding synthesis filter coefficients are given in Table 1. The default *reversible transformation* is implemented by means of the 5-tap/3-tap filter [10], the coefficients of which are given in Table 2.

The standard supports two filtering modes: *convolution-based* and *lifting-based*.[11] Both modes require that the signal should be extended periodically. This *periodic symmetric extension* is used to ensure that for the filtering operations that take place at the boundaries of the signal, a signal sample exists and spatially corresponds to each coefficient of the filter mask. The number of additional samples required at the boundaries of the signal is therefore filter-length dependent.

Table 1. Daubechies 9/7 analysis and synthesis filter coefficients

	Analysis Filter Coefficients		Synthesis Filter Coefficients	
i	Lowpass Filter $h_L(i)$	Highpass Filter $h_H(i)$	Lowpass Filter $g_L(i)$	Highpass Filter $g_H(i)$
0	0.6029490182363579	1.115087052456994	1.115087052456994	0.6029490182363579
±1	0.2668641184428723	-0.5912717631142470	0.5912717631142470	-0.2668641184428723
±2	-0.07822326652898785	-0.05754352622849957	-0.05754352622849957	-0.07822326652898785
±3	-0.01686411844287495	0.09127176311424948	-0.09127176311424948	0.01686411844287495
±4	0.02674875741080976			0.02674875741080976

Table 2. 5/3 analysis and synthesis filter coefficients

	Analysis Filter Coefficients		Synthesis Filter Coefficients	
i	Lowpass Filter $h_L(i)$	Highpass Filter $h_H(i)$	Lowpass Filter $g_L(i)$	Highpass Filter $g_H(i)$
0	6/8	1	1	6/8
±1	2/8	-1/2	1/2	-2/8
±2	-1/8			-1/8

Convolution-based filtering consists in performing a series of dot products between the two filter masks and the extended 1-D signal. *Lifting-based filtering* consists of a sequence of very simple filtering operations for which alternately odd sample values of the signal are updated with a weighted sum of even sample values, and even sample values are updated with a weighted sum of odd sample values. For the reversible (lossless) case the results are rounded to integer values. The lifting-based filtering for the 5/3 analysis filter is achieved by means of eq. (1) below:[2]

$$y(2n+1) = x_{ext}(2n+1) - \left\lceil \frac{x_{ext}(2n) + x_{ext}(2n+2) - 1}{2} \right\rceil \qquad (1a)$$

$$y(2n) = x_{ext}(2n) + \left\lfloor \frac{y(2n-1) + y(2n+1) + 2}{4} \right\rfloor, \qquad (1b)$$

where x_{ext} is the extended input signal, y is the output signal and $\lfloor a \rfloor, \lceil a \rceil$ indicate the largest integer not exceeding *a* and the smallest integer not exceeded by *a*, respectively.

The 5/3 filter allows repetitive encoding and decoding of an image without any additional loss, if there is no change in the compression factor and if the decompressed image values are not clipped when they fall outside the full dynamic range (for example 0-255 for an 8 bpp image). If the original image did not use the full dynamic range (for example it had only values from 20-200), then clipping is not a problem. If clipping occurs, it can cause successive loss with each re-compression (even if the quantisation step remains the same).

3.2. Quantisation

After transformation, all coefficients are quantised. Quantisation is the process by which the coefficients are reduced in precision. This operation is lossy, unless the quantisation step is 1 and the coefficients are integers, as produced by the reversible integer 5/3 wavelet. Each of the transform coefficients $a_b(u,v)$ of the subband b is quantised to the value $q_b(u,v)$ according to the formula:

$$q_b(u,v) = sign(a_b(u,v)) \left\lfloor \frac{|a_b(u,v)|}{\Delta_b} \right\rfloor, \qquad (2)$$

where Δ_b is the quantisation step size. One quantisation step size per subband is allowed. All quantised transform coefficients are signed values even when the original components are unsigned. These coefficients are expressed in a sign-magnitude representation prior to coding. For reversible compression, the quantisation step size is required to be 1.

3.3. Entropy Coding

Each subband of the wavelet decomposition is divided into rectangular blocks, called *code-blocks*.[12] These are the fundamental entities for entropy coding, i.e. each code-block is independently coded (using arithmetic coding) with its own embedded bit-stream. A number of spatially consistent code-blocks from each subband at a given resolution level constitute a so-called *precinct* (Fig. 3). Code-blocks are coded a bit-plane at a time, starting with the most significant bit-plane with a non-zero element. For each bit-plane in a code-block, a special code-block scan pattern is used for each of three passes (significance propagation pass, magnitude refinement pass and clean-up pass).[2 5] Each coefficient bit in the bit-plane is coded in only one of the three passes. A rate distortion optimization method is used to allocate a certain number of bits to each block. In *arithmetic coding*, with each binary decision the current probability interval is subdivided into two sub-intervals, and the codestream is modified (if necessary) so that it points to the base (the lower bound) of the probability sub-interval assigned to the symbol which occurred. Since the coding process involves addition of binary fractions rather than concatenation of integer codewords, the more probable binary decisions can often be coded at a cost of much less than one bit per decision.

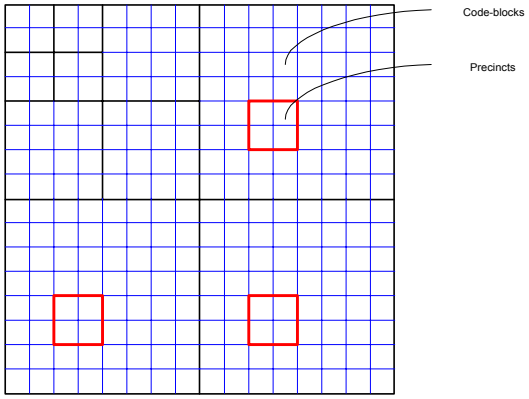

Figure 3. Tile partition into subbands, precincts and code-blocks

4. EXPERIMENTAL RESULTS

All experiments have been conducted using the JPEG2000 VM7.0 software.[13] A variety of computer platforms (HP, Sun, PC) and operating systems (UNIX, LINUX, DOS) were used. The test images are given in Table 3. The *gprof* tool was used in the UNIX and LINUX systems. All images were compressed and expanded at six different target bit-rates, namely 2, 1, 0.5, 0.25, 0.125, 0.0625 bpp. The dyadic decomposition and the reversible 5/3 integer filter were used in all cases. Color images were both reversibly and irreversibly color transformed. Different options were used in order to explore their influence on memory usage and execution time, as shown in Table 4. In the first row of Table 4 we used the defaults for the tiles (i.e. no tiles) and the code-block size (i.e. square of size 64). We then modified the code-block size (reducing it down to 32x32), the mode of coding (employing the 'lazy coding mode'), and the type of compression (lossy or lossless). In the 'lazy coding mode' the arithmetic coding of some of the bit-planes is bypassed. Results were fairly similar for all test images on all computer platforms, thus only those concerning the image 'woman' are given in the following.

Table 3. Test images

Image	Size	Bit-Depth	Type
Lenna	512 x 512	8 bit	grayscale
Target	512 x 512	8 bit	grayscale
Gold	720 x 576	8 bit	grayscale
Hotel	720 x 576	8 bit	grayscale
Ski	720 x 576	24 bit	color
Woman	2048 x 2560	24 bit	color

Table 4. Test cases

	VM7.0 Options					
	-Fkernels W5x3	-Mycc	-Cblk 32	-Clazy	-Frev	-Ftiles 128x128
1	•	•				
2	•	•	•			
3	•	•		•		
4	•	•	•	•		
5	•	•			•	
6	•	•	•		•	
7	•	•		•	•	
8	•	•	•	•	•	
9	•	•				•

In Fig. 4 the memory usage for all cases for the encoder and the decoder is depicted. From these graphs we can easily deduce that:

a. Memory usage is independent of the bit rate (encoder).
b. Memory usage increases as the bit rate increases (decoder).
c. Decoder uses less memory (by almost an order of magnitude) than the corresponding encoder does.
d. The decrease in the code-block size leads to an increase in memory usage by approximately 20% (encoder).
e. Lazy coding mode reduces memory requirements (encoder).
f. Lossless (reversible) coding results in an increase by approximately 25% to the memory usage as compared to the lossy (irreversible) coding (encoder).

The execution time (i.e. time per pixel) graphs for the encoder and the decoder are shown in Fig. 5. It is seen that:

g. Execution time is almost independent of the bit-rate (encoder).
h. Execution time increases almost linearly with bit-rate (decoder).
i. The decrease in the code-block size results in a slight increase of the execution time.
j. The lazy coding mode speeds-up the encoder by approximately 20%.

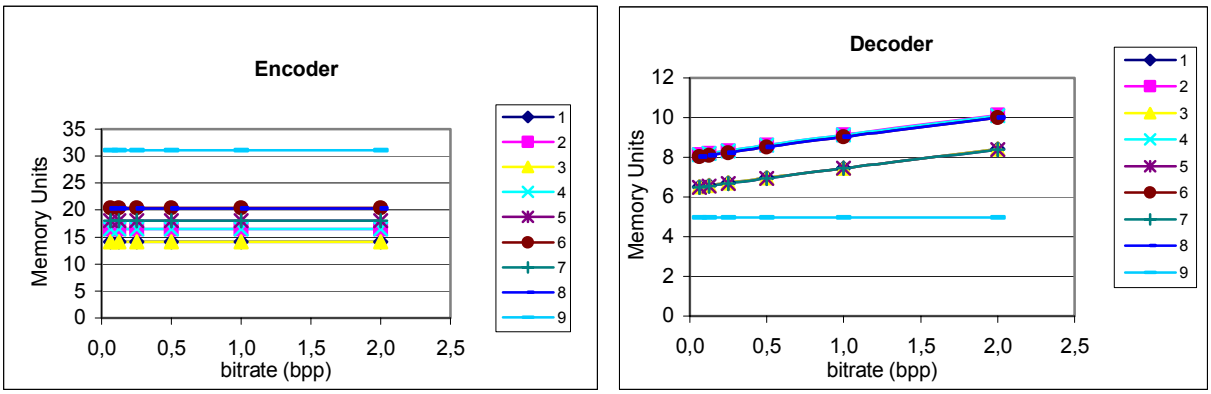

Figure 4. Memory usage: (a) encoder and (b) decoder.

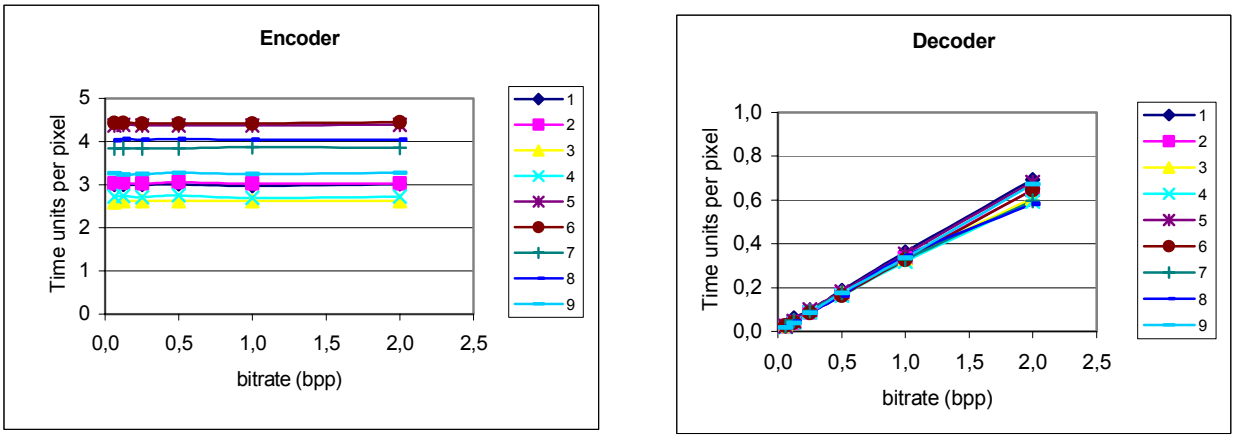

Figure 5. Execution time: (a) encoder and (b) decoder

From the above listed points we can conclude that:

I. There is a direct relation between memory usage (i.e. memory accesses) and execution time (points a, b and g, h respectively).

II. The fact that memory usage and execution time for the encoder does not depend on bit rate is due to the dominance of the DWT and the entropy coding cost. The transform does not change with bit rate and the coding of the rather finely quantised coefficients is achieved by a three-pass process and a subsequent truncation of embedded bit streams to meet the precise target bit rate. Most of the complexity of the JPEG2000 encoder / decoder implementation (over 80%) lies within the DWT and the entropy coder / decoder.

III. The increase of the execution time with the decrease of the code-block size is mostly due to the miscellaneous functions for handling the blocks.

IV. The partition into tiles results in a considerable increase of the memory usage of the encoder. This is in contradiction with the main concept for introducing tile partition, but it is only due to the generality of the particular software implementation, which has been developed for testing purposes. A carefully optimized implementation will resolve all these problems and will reveal the benefits of tile and block partitions.

5. CONCLUSIONS

In this communication, a practical analysis of the complexity of the JPEG2000 encoder and decoder implementation on different computer platforms has been presented. The VM7.0 software (in C) was used. It has been verified that the discrete wavelet transform and the entropy coding are the most time and memory consuming parts of the algorithm. Thus, any attempt to optimize these parts will substantially improve the overall implementation. In addition, it has been seen that the encoder's implementation complexity is almost independent of the bitrate. For the decoder, however, memory and time complexity is in direct relation to bitrate. As expected, the decoder is less demanding than the encoder and the lazy mode results in a considerable speed up of both the encoder and the decoder.

REFERENCES

1. ISO/IEC JTC1/SC29/WG1 N505, *Call for contributions for JPEG 2000 (JTC 1.29.14, 15444): Image Coding System*, March 1997.
2. M. Boliek, C. Christopoulos and E. Majani (editors), *JPEG2000 Part I Final Committee Draft version 1.0*, ISO/IEC JTC1/SC29/WG1 N1646R, March 2000.
3. ISO/IEC JTC1/SC29/WG1 N1271, *JPEG2000 Requirements and Profiles v5.0*, March 1999.
4. C. Christopoulos, J. Askelof and M. Larsson, *Efficient methods for encoding regions of interest in the upcoming JPEG2000 still image coding standard*, IEEE Signal Processing Letters, Sep. 2000.
5. M. W. Marcellin, M. Gormish, A. Bilgin and M. Boliek, *An Overview of JPEG 2000*, Proc. of Data Compression Conference, Snowbird, Utah, pp. 523-541, March 2000.

6. A. Skodras, C. Christopoulos and T. Ebrahimi, *JPEG2000: The Upcoming Still Image Compression Standard*, Invited Tutorial in the 11th Portuguese Conference on Pattern Recognition (RECPAD 2000), Porto, Portugal, May 11-12, 2000 (also available on http://etro.vub.ac.be/~chchrist/).
7. C. Christopoulos and A. Skodras, *The JPEG 2000*, Tutorial in IEEE International Conference on Image Processing (ICIP 99), October 25-28, 1999, Kobe, Japan (also available on http://etro.vub.ac.be/~chchrist/).
8. G. K. Wallace, *The JPEG Still Picture Compression Standard*, IEEE Trans. Consumer Electronics, Vol. 38, No 1, pp. xviii-xxxiv, Feb. 1992.
9. M. Antonini, M. Barlaud, P. Mathieu and I. Daubechies, *Image Coding Using the Wavelet Transform*, IEEE Trans. Image Proc., pp. 205-220, April 1992.
10. A. R. Calderbank, I. Daubechies, W. Sweldens and B.-L. Yeo, *Lossless Image Compression Using Integer to Integer Wavelet Transforms,* Proc. IEEE International Conference on Image Processing (ICIP 97), Santa Barbara, USA, vol. 1, pp. 596-599, Oct. 1997.
11. J. Kovacevic and W. Sweldens, *Wavelet Families of Increasing Order in Arbitrary Dimensions*, IEEE Trans. on Image Processing, Vol. 9, No. 3, pp. 480-496, March 2000.
12. D. Taubman, *High Performance Scalable Image Compression with EBCOT*, Proc. IEEE Int. Conference on Image Processing (ICIP 99*)*, Vol.III, pp. 344-348, 24-28 October 1999, Kobe, Japan.
13. C. Christopoulos, *JPEG2000 Verification Model 7.0 (technical description),* ISO/IEC JTC1/SC29/WG1 N1684, April 25, 2000.