JasPer: A Software-Based JPEG-2000 Codec Implementation*

Michael D. Adams and Faouzi Kossentini
Dept. of Elec. and Comp. Engineering, University of British Columbia
Vancouver, B.C., Canada V6T 1Z4
mdadams@ece.ubc.ca and faouzi@ece.ubc.ca

ABSTRACT

A software-based implementation of the image codec specified in the emerging JPEG-2000 standard is discussed. The run-time complexity and coding performance of this implementation are also briefly analyzed.

1. INTRODUCTION

Digital imagery is pervasive in our world today. Hence, standards for the efficient representation and interchange of such information are essential. JPEG 2000 [1] is a new image compression standard currently being developed by the International Organization for Standardization (ISO). In this paper, we describe JasPer, one of the first implementations of this newly emerging standard. We also analyze the run-time complexity and coding performance of the implementation. By studying code execution profiles for typical coding scenarios, we gain improved insight into how one might develop a fast software implementation. Also, we examine the impact of using fixed-point (instead of floating-point) arithmetic on coding performance.

2. JPEG 2000

JPEG 2000 supports lossy and lossless compression of single-component (e.g., grayscale) and multi-component (e.g., color) imagery. In addition to this basic compression functionality, however, numerous other features are provided, including: 1) progressive recovery of an image by fidelity or resolution; 2) region-of-interest coding, whereby different parts of an image can be coded with differing fidelity; 3) random access to specific regions of an image without needing to decode the entire code stream; 4) a flexible file format; and 5) good error resilience. Due to its excellent coding performance and many attractive features, there is a very large potential application base for JPEG 2000. Some possible application areas include: image archiving, web browsing, document imaging, digital photography, medical imaging, and remote sensing.

3. JPEG-2000 CODEC

The JPEG-2000 codec handles both lossy and lossless compression using the same transform-based framework, and borrows heavily on ideas from [2]. The general structure of the encoder is shown in Figure 1(a). Suppose that we have an image with one or more components. Let P denote the number of bits per sample for a particular component. The sample values may be either signed or unsigned, leading to a nominal dynamic range of $[-2^{P-1}, 2^{P-1}-1]$ or $[0, 2^P-1]$, respectively. The encoding process proceeds as follows:

1) *Forward multicomponent transform.* If a component has unsigned sample values, these values are adjusted by subtracting a bias of 2^{P-1}, before any further processing takes place. If the image has three components, one of two multicomponent transforms may, at the discretion of the encoder, be applied to the image data. Such a transformation is intended to map image data from the (commonly used) RGB color space to the YCrCb color space. In the case of lossy coding, the transform employed (i.e., the ICT [1]) is nonreversible and real-to-real. In the case of lossless coding, the transform used (i.e., the RCT [1]) is reversible and integer-to-integer.

2) *Forward wavelet transform.* A wavelet transform may optionally be applied to each component. All supported transforms are based on 1-D 2-channel filter banks. In order to handle 2-D data, 1-D transforms are applied separably in each dimension. Symmetric extension is employed to handle finite-length signals. In the lossy case, a nonreversible real-to-real wavelet transform is used (i.e., the 9/7 transform of [3]). In the lossless case, a reversible integer-to-integer transform is employed (i.e., the 5/3 transform of [4]).

3) *Quantization.* The transform coefficients are quantized with a deadzone scalar quantizer. A different quantizer is employed for the transform coefficients of each subband. In the case of lossless coding, the quantizer step sizes are forced to be one, so that the quantizer indices are nothing more than the transform coefficients themselves.

4) *Tier-1 encoder.* The quantizer indices associated with each subband are partitioned into fixed-size code blocks (typically 64×64). Then, an embedded code stream is produced for each block independently using bit-plane coding.

The bit-plane coding scheme functions as follows. A block of samples (i.e., quantizer indices) is treated as a small image. Then, this small image is coded one bit plane at a time, starting with the most significant bit plane. There are three passes per bit plane. The three passes are defined as follows:

i) *Significance pass.* This pass encodes the significance of any sample that has not yet been found to be significant and is predicted to become significant during the processing of the current bit plane. If a sample becomes significant, its sign is also encoded in this pass.

ii) *Refinement pass.* This pass encodes the next most significant bit for any sample that has been found significant during the processing of a previous bit plane.

iii) *Cleanup pass.* This pass encodes the significance of any sample that has not yet been found to be significant and was

*THIS WORK WAS SUPPORTED BY BOTH THE NATURAL SCIENCES AND ENGINEERING RESEARCH COUNCIL OF CANADA, AND IMAGE POWER INC.

predicted to remain insignificant in the preceding significance pass. If a sample becomes significant, its sign is also encoded in this pass.

The symbols produced by the bit-plane encoder are coded using an adaptive binary arithmetic coder (i.e, the MQ coder employed in the JBIG2 standard [5]). Optionally, arithmetic coding can be bypassed for some of the symbols produced during the processing of the lesser significant bit planes.

5) *Tier-2 encoder.* The bit-plane coding passes that are to be included for each code block and the order of appearance of these passes in the final code stream are encoded along with the actual coding pass data. For lossless coding, all bit-plane coding passes are included in the code stream. For lossy coding, only a subset of the coding passes are typically included.

Rate control is achieved through both the choice of quantizer step sizes and also the selection of the subset of coding passes to include in the final code stream.

The structure of the decoder, shown in Figure 1(b), essentially mirrors that of the encoder. The decoding process functions much like the encoding process run backwards:

1) *Tier-2 decoder.* The bit-plane coding passes for the various code blocks are extracted from the code stream. Except in the lossless case, not all coding passes for all code blocks are guaranteed to be present.

2) *Tier-1 decoder.* The bit-plane coding passes for each of the code blocks is decoded, yielding the reconstructed quantizer indices. In the lossy case, not all of the bit-plane coding passes are typically present in the code stream, in which case the reconstructed quantizer indices are themselves only approximations to the original quantizer indices.

3) *Dequantization.* The quantized transform coefficient values are obtained from the reconstructed quantizer indices. In the case of lossless coding, the transform coefficients are the same as the quantizer indices.

4) *Inverse wavelet transform.* The inverse wavelet transform is applied to the data for each component (if applicable).

5) *Inverse multicomponent transform.* The inverse multicomponent transform is applied to the image data (if applicable). If the sample values for a component are unsigned, the original nominal dynamic range is restored by adding a bias of 2^{P-1}. In the case of lossy coding, a clipping operation is performed on the sample values to ensure that they do not exceed their allowable range.

4. JASPER

JasPer is a software-based implementation of the codec specified in the emerging JPEG-2000 standard. The development of this software had two motivations. First, we wanted to develop a JPEG-2000 implementation using the working draft [6] as our only reference. This would allow inaccuracies and ambiguities in the working draft to be identified and corrected. Second, by conducting interoperability testing with other JPEG-2000 implementations, we might find further ambiguities in the text of the working draft, allowing them to be corrected.

The design of the JasPer software was driven by several key concerns: fast execution speed, efficient memory usage, robustness, portability, modularity, maintainability, and extensibility. Since fixed-point operations are typically faster than their floating-point counterparts on most platforms, and some platforms lack hardware support for floating-point operations altogether, we elected to use only fixed-point operations in our software. By making such a choice, we are better able to meet our objectives of high portability and fast execution speed.

The JasPer software is written in the C programming language. This language was chosen mainly due to the availability of C development environments for most of today's computing platforms. The JasPer software consists of about 20k lines of code in total. This code is spread across several libraries. There are two executable programs, the first being the encoder, and the second being the decoder. The JasPer software can handle image data in a number of popular formats (e.g., PGM/PPM, Windows BMP, and Sun Rasterfile).

Although the JasPer software is intended to support all of the functionality described in Part 1 of the JPEG-2000 standard, more work is still required in order to achieve this goal. In the current version of the software, there are a few missing/incomplete functionalities, most of which relate to region of interest coding or error resilience. These missing functionalities will be added in the next release of the software.

5. CODE EXECUTION PROFILE

When implementing a complex algorithm in software, it is often beneficial to know which parts of the algorithm are likely to contribute the most to total execution time. With this as motivation, we have performed some basic profiling of our codec software, the results of which are presented below.

For evaluation purposes, three test images were employed, as listed in Table 1. The test suite was chosen to include grayscale, color, natural, and synthetic imagery. Two typical coding scenarios were considered: one lossless and one lossy. In each case, we measured the execution time associated with each of the basic encoder/decoder operations (i.e., multicomponent transform, wavelet transform, quantization, tier-1 coding, tier-2 coding, and rate control).

Table 1. Test images

Image	Characteristics	Description
`target`	grayscale, 512×512, 8 bpp/component	patterns and textures [7]
`mat`	grayscale, 1528×1146, 8 bpp/component	mountains [7]
`announcer`	RGB color, 512×480, 8 bpp/component	woman

5.1. Lossless Coding

In the first coding scenario, the three test images were encoded in a lossless manner, and then the resulting code streams were decoded. In each case, the encoder and decoder execution was profiled, leading to the results shown in Table 2.

First, let us consider the case of the encoder. Examining Table 2(a), we see that the encoder spends most of its time on tier-1 coding (typically more than 50%), followed next by the wavelet transform. In fact, these two operations together account for the

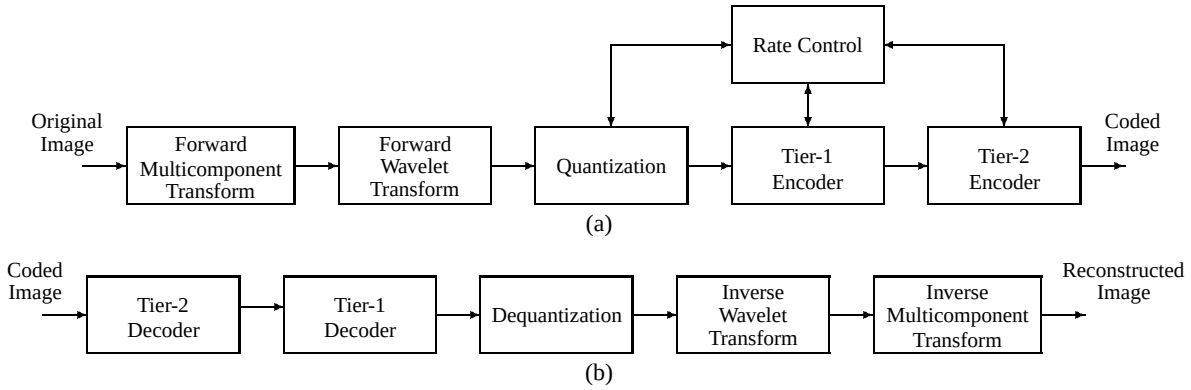

Fig. 1. Structure of the JPEG-2000 codec. The structure of the (a) encoder and (b) decoder.

vast majority of the encoder's execution time (typically 85-90%). The amount of time consumed by the multicomponent transform and tier-2 coding operations is much less significant. Examining tier-1 coding in more detail, we observe that for natural imagery, significance coding usually requires the most time, followed by the cleanup coding, and then refinement coding. For synthetic imagery, however, this trend is not necessarily observed, as evidenced by the results for the `target` image. Thus, the execution time breakdown for tier-1 coding is somewhat sensitive to image content.

Now, let us consider the case of the decoder. The execution profiling results for the decoder are given in Table 2(b). Again, we can see that the decoder spends most of its time on tier-1 coding (typically 50% or more), followed next by the wavelet transform. These two operations combined account for the vast majority of the decoder's execution time (typically 80-85%). The multicomponent transform and tier-2 coding operations consume relatively little time. Examining the tier-1 coding process in more detail, we observe similar trends as in the encoder. That is, for natural imagery, significance coding requires the most time, followed by cleanup coding, and then refinement coding, while the breakdown is less predictable for synthetic imagery.

5.2. Lossy Coding

In the second coding scenario, the three test images were encoded in a lossy manner, and then the resulting code streams were decoded. Again, in each case, the encoder and decoder execution was profiled. This led to the results given in Table 3.

First, let us study the results for the encoder, given in Table 3(a). Clearly, the tier-1 coding and wavelet transform operations require the most time. Together, these two operations account for the vast majority of the execution time (typically 75-85%). The multicomponent transform, quantization, tier-2 coding, and rate control operations require much less time. Looking at the tier-1 coding process in more detail, we observe that the most time is usually required by cleanup coding, followed by significance coding, and then refinement coding.

In the JasPer encoder, rate control is achieved through the selection of the coding passes to include in the final code stream. The quantizer step sizes are fixed. To ensure optimal rate-distortion performance, the encoder generates all of the coding passes during tier-1 coding (even though some coding passes may not be included in the final code stream). This leads to the large amount of time consumed by tier-1 coding in the encoder. Since quantization takes place, however, the tier-1 coding overhead is not as great (in absolute terms) as in the lossless case. Also, different wavelet transforms are employed in the lossy and lossless cases. The 9/7 wavelet transform employed in the lossy case requires more computation, helping to contribute to the larger amount of time spent on the wavelet transform operation relative to the lossless case.

Now, let us consider the case of the decoder. From the results in Table 3(b), we can see that the wavelet transform clearly requires the most time by far (typically 60-80%). Tier-1 coding is the next largest time consumer, but has much more modest requirements (typically 5-10%). The multicomponent transform, quantization, and tier-2 coding operations require relatively little time. Looking at the tier-1 coding process in more detail, we observe that, for natural imagery, the least time is required by refinement coding. Generally speaking, the execution time breakdown for tier-1 coding depends, to a large extent, on the particular image and bit rate employed.

Table 2. Codec execution profile for lossless coding (1 tile, SNR progressive, 1 layer, 5/3 transform, 5 decomposition levels). (a) Encoder. (b) Decoder.

(a)

Operation	% of Total Execution Time		
	`target`	`mat`	`announcer`
multicomponent transform	—	—	1.3
wavelet transform	36.5	24.1	31.0
tier-1 coding	53.9	62.1	53.6
significance	12.4	23.9	22.8
refinement	18.5	15.0	11.4
cleanup	19.7	19.0	17.5
tier-2 coding	1.1	1.8	1.4

(b)

Operation	% of Total Execution Time		
	`target`	`mat`	`announcer`
multicomponent transform	—	—	1.5
wavelet transform	35.9	25.3	34.4
tier-1 coding	47.8	56.9	48.2
significance	17.0	24.2	24.9
refinement	19.8	12.4	8.8
cleanup	10.3	19.1	12.6
tier-2 coding	2.2	1.5	1.5

Table 3. Codec execution profile for lossy coding (32:1 compression for grayscale images and 64:1 compression for color images, 1 tile, SNR progressive, 1 layer, 9/7 transform, 5 decomposition levels). (a) Encoder. (b) Decoder.

(a)

Operation	% of Total Execution Time		
	`target`	`mat`	`announcer`
multicomponent transform	—	—	3.1
wavelet transform	49.1	40.0	54.8
quantization	2.3	3.2	2.2
tier-1 coding	34.5	40.9	23.6
significance	11.2	13.9	8.1
refinement	8.7	7.1	3.6
cleanup	12.6	15.9	9.5
tier-2 coding	0.1	0.1	0.1
rate control	2.9	4.6	3.5

(b)

Operation	% of Total Execution Time		
	`target`	`mat`	`announcer`
multicomponent transform	—	—	2.6
wavelet transform	75.0	63.5	75.4
quantization	2.5	3.3	1.7
tier-1 coding	8.0	9.9	4.9
significance	3.0	3.9	0.8
refinement	3.1	0.9	0.1
cleanup	1.4	3.1	2.9
tier-2 coding	0.1	0.3	0.1

6. CODING PERFORMANCE

In this section, we briefly evaluate the performance of the the JasPer software. To this end, we compare our implementation to the JPEG-2000 verification model (VM) software (version 5.2) which is also written in the C programming language. In particular, we compare the coding performance of the decoders for lossy compression.

To begin, the three test images from the previous section were encoded in a lossy manner at several bit rates. Both the JasPer and VM decoders were then used to decode each of these code streams. The distortion for each of the reconstructed images was then measured. Before proceeding further, it is important to note that the VM software uses floating-point arithmetic, while the JasPer software uses fixed-point arithmetic exclusively. Thus, one might expect the coding performance of the JasPer decoder to be worse than that of the VM decoder. Table 4 gives the results for several pairs of test images and bit rates. Examining the numbers, we observe that both decoders yield comparable results. Moreover, subjective image quality is also comparable. These results demonstrate that, provided one is careful, it is possible to implement the decoder in fixed-point arithmetic without incurring a large distortion penalty.

7. CONCLUSIONS

JasPer, a software-based implementation of the image codec specified in the emerging JPEG-2000 standard has been introduced. The initial version of the JasPer software performs well in terms coding performance. By examining code execution profiles, we provided some better insight into where a typical software implementation is likely to spend most of its time executing. These results demonstrated the importance of efficient wavelet transform and tier-1 coding engines. Also, we briefly examined the impact of using fixed-point (instead of floating-point) arithmetic. We demonstrated that, provided sufficient care is taken, the use of fixed-point arithmetic need not introduce any significant penalty in coding performance.

Table 4. Comparison of decoder performance for lossy coding (1 tile, SNR progressive, 1 layer, 5 decomposition levels)

Image	CR†	PSNR (dB)	
		JasPer	VM 5.2
`target`	64:1	23.75	23.74
	32:1	28.01	27.99
	16:1	34.92	34.88
`mat`	64:1	33.83	33.84
	32:1	36.96	36.98
	16:1	40.78	40.91
`announcer`	128:1	32.01,32.81,30.79	32.00,32.82,30.82
	64:1	34.91,35.94,33.46	34.94,35.97,33.50
	32:1	37.75,38.74,35.87	37.86,38.83,35.96

† compression ratio

8. SOFTWARE AVAILABILITY

The JasPer software is available from the web sites of the Signal Processing and Multimedia Group at the University of British Columbia (i.e., `http://spmg.ece.ubc.ca`) and Image Power Inc. (i.e., `http://www.imagepower.com`). This software has also been submitted to the ISO for use as a reference implementation in Part 5 of the JPEG-2000 standard. For more information about JasPer, the reader is referred to the first author's home page: `http://www.ece.ubc.ca/~mdadams`.

9. REFERENCES

[1] ISO/IEC, *ISO/IEC FCD15444-1, Information technology - JPEG 2000 image coding system*, Mar. 2000, Available from `http://www.jpeg.org`.

[2] D. Taubman, "High performance scalable image compression with EBCOT," in *Proc. of IEEE International Conference on Image Processing*, Kobe, Japan, 1999, vol. 3, pp. 344–348.

[3] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. on Image Processing*, vol. 1, no. 2, pp. 205–220, Apr. 1992.

[4] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Wavelet transforms that map integers to integers," *Applied and Computational Harmonic Analysis*, vol. 5, no. 3, pp. 332–369, July 1998.

[5] ISO/IEC, *ISO/IEC 14492-1, Lossy/lossless coding of bi-level images*, 2000.

[6] ISO/IEC, *ISO/IEC 15444 Working Draft (Version 3.0)*, Nov. 1999.

[7] "JPEG-2000 test images," ISO/IEC JTC 1/SC 29/WG 1 N 545, July 1997.