A Prototype Distributed Visualization System

Part II: ERIC (Efficient Reversible Image Compression)

Contents

1 Introduction

This section describes a new lossless/lossy and progressive image compression/decompression algorithm. This algorithm will be referred to as ERIC, which stands for "Efficient Reversible Image Compression."

This algorithm losslessly compresses an image into multiple segments. Each segment provides information about the image at a corresponding level of spatial resolution and amplitude accuracy. Decompression of the first segment reconstructs the image at its lowest accuracy level. Decompression of further segments of the compressed image reconstructs the image at progressively better levels of accuracy. Decompression of all segments reconstructs the original image exactly.

This algorithm is particularly useful in bandwidth-constrained storage/retrieval applications when the user requires the closest approximation to the original image, for a given bandwidth constraint. Less accurate approximations take less bandwidth to transmit while allowing the user to make a decision as to whether the full resolution is desired or not, minimizing total transmission time per decision.

This algorithm is a progressively lossless compression/decompression algorithm and requires no additional storage since all successive approximations of the image can be reconstructed directly from the compressed image bit stream. Furthermore, the lossless compression ratio obtained with this algorithm is comparable to or better than the best standard lossless compression algorithms, which do not allow for progressively accurate approximations. Finally, the wavelet transform used for decorrelating image pixels has a very fast integer implementation, requiring only 5 additions and 2 shifts per image pixel.

Section 2 describes the compression algorithm, while Section 3 describes the decompression algorithm. Section 4 evaluates the performance of the algorithm described.

2 Compression Algorithm

At compression, a digital image is transformed by means of a new perfect reconstruction (PR) integer wavelet transform (IWT) into two lower resolution images, a low-frequency and a high-frequency image. It is an IWT in that it transforms integer input sequences into integer output sequences of the same length, and it is PR in that the IWT is reversible; i.e., the original image can be recovered exactly. This transform procedure is most often "entropy-reducing" in that the entropy of the transform image is lower. The high-frequency image is entropy coded, while the low-frequency image may either be entropy coded, or further transformed by the IWT. Entropy coded segments, starting with the ones associated with the lowest resolution levels, make up the compressed image.

The compression algorithm takes place in two steps:

- forward IWT: this step is entropy-reducing and decomposes the original image into subimages containing information at various levels of spatial resolution; and
- entropy coding: the subimages are entropy coded separately.

2.1 Notation

Given that i is an integer and x a real number on the half-open intervals below, we define two operators: for rounding down ($\lfloor \ldots \rfloor$), and for rounding up ($\lceil \ldots \rceil$).

$$\lfloor x \rfloor = i \Leftrightarrow x \in \ (i - \frac{1}{2}, i + \frac{1}{2}]$$

$$\lceil x \rceil = i \Leftrightarrow x \in \ [i - \frac{1}{2}, i + \frac{1}{2})$$

2.2 Forward IWT

Independent of the IWT and the image's dimensions, the procedure to compute the entropy-reducing forward IWT is always the same. One level of a forward IWT decomposes an image X into a low-frequency subimage X_0 and a high-frequency subimage X_1, both at lower resolutions. The subimage X_0 is a lower resolution version of image X and can be used along with image X_1 to reconstruct the original image X. This decomposition represents one level of the forward IWT. After any level of decomposition, normalization of transform coefficients is necessary, as described later.

Typically, several levels of the forward IWT (a good default number is four) are computed, by iterating the procedure just described to the low-frequency subimage produced by the last decomposition. For example, the two-level forward IWT would be obtained by further decomposing the low-frequency subimage X_0 into subimages X_{00} and X_{01}. Therefore, an n-level forward IWT transforms an input image X into $(n + 1)$ subimages, one of which corresponds to the low-frequency subimage at the lowest level of spatial resolution, and n high-frequency subimages at all levels of spatial resolution.

In the following paragraphs, we describe the computation of only one level of the forward IWT, as it is computed for images.

2.2.1 One-Dimensional Transform

Given an integer input sequence $(x_i)_{i=0,\ldots,N-1}$, its forward IWT $(y_i)_{i=0,\ldots,N-1}$, will also be an integer sequence, which will be computed depending on whether the length N of the integer sequence is even or odd.

If the signal length N is even (i.e., $N = 2K$), then the integer transform sequence is computed in the following two steps, odd coefficients first

$$\begin{cases} y_{2i+1} = x_{2i+1} - \lfloor (x_{2i} + x_{2i+2})/2 \rfloor & \text{for} \quad i = 0, \ldots, K-2 \\ y_{2i+1} = x_{2i+1} - x_{2i} & \text{for} \quad i = K-1 \end{cases}$$

and even coefficients second

$$\begin{cases} y_{2i} = x_{2i} + \lfloor y_{2i+1}/2 \rfloor & \text{for} \quad i = 0 \\ y_{2i} = x_{2i} + \lfloor (y_{2i-1} + y_{2i+1})/4 \rfloor & \text{for} \quad i = 1, \ldots, K-1 \end{cases}$$

If the signal length N is odd (i.e., $N = 2K + 1$), then the integer transform is computed in the following two steps, odd coefficients first

$$y_{2i+1} = x_{2i+1} - \lfloor (x_{2i} + x_{2i+2})/2 \rfloor \ \text{for} \quad i = 0, \ldots, K-1$$

and even coefficients second

$$\begin{cases} y_{2i} = x_{2i} + \lfloor y_{2i+1}/2 \rfloor & \text{for } \ i = 0 \\ y_{2i} = x_{2i} + \lfloor (y_{2i-1} + y_{2i+1})/4 \rfloor & \text{for } \ i = 1, \ldots, K-1 \\ y_{2i} = x_{2i} + \lfloor y_{2i-1}/2 \rfloor & \text{for } \ i = K \end{cases}$$

2.2.2 Two-Dimensional Transform

One level of a forward IWT for an image X can be computed in two steps:

- one one-level forward IWT in the horizontal dimension, producing two integer output subimages,
- two one-level forward IWTs in the vertical dimension, producing four integer output subimages.

The one-level forward IWT in the horizontal dimension produces one low-frequency subimage Y_0 and one high-frequency subimage Y_1. The one-level forward IWT in the vertical dimension is applied to *both* subimages Y_0 and Y_1, and produces therefore four subimages: one low-frequency subimage Y_{00_1} and three high-frequency subimages Y_{01_1}, Y_{10_1}, and Y_{11_1}. The second level of the IWT would replace Y_{00_1} with Y_{00_2}, Y_{01_2}, Y_{10_2}, and Y_{11_2} (see Figure 1). Additional levels of decomposition are always performed on the subimage of lowest-frequency coefficients.

2.2.3 Dropping High-Frequency Transform Coefficients

A simplified but non-reversible IWT can be computed, to reduce computation time; this is done by dropping high-frequency transform coefficients (i.e., using y_{2i+1} to compute y_{2i} as described above, and then setting y_{2i+1} to zero). This procedure corresponds to low-pass filtering and downsampling the image, thereby diminishing the image size by a factor of four. One common use of this simplified IWT is found when lossily compressing an image at a very high compression ratio: at the first level, the simplified IWT is used, followed by the usual reversible IWT described above. The number of times the simplified IWT is used is referred to as nl_{low}, while the subsequent number of regular levels of decomposition is referred to as nl_{lh}.

2.3 Successive Approximation of Integers

We now describe the binary representation for the successive approximation of integers. In the following discussion, the variable l signifies the number of levels of approximation sought, as defined in Section 2.6.

Given an integer $x \in [-(2^n - 1), 2^n - 1]$, it can uniquely be represented by its sign (if it is nonzero) and the binary representation of its absolute value:

$$x = \mathrm{sign}(x) \cdot (\sum_{i=0}^{n-1} b_i 2^i)$$

Table 1: Successive Approximation of Integers

x	sign	b_1	b_0
-3	-(1)	1	1
-2	-(1)	1	0
-1	-(1)	0	1
0		0	0
1	+(0)	0	1
2	+(0)	1	0
3	+(0)	1	1

Table 2: Binary Representation for Successive Approximation

x	b_1	b_0
-3	1^1	1
-2	1^1	0
-1	0	1^1
0	0	0
1	0	1^0
2	1^0	0
3	1^0	1

with $\text{sign}(x) = \pm 1$ for $x \neq 0$ and $b_i \in \{0, 1\}$. This amounts to an $(n+1)$–bit binary representation for nonzero integers and an $n-$bit representation for $x = 0$ (see Table 1 for $n = 2$).

Since we are interested in a binary representation which easily lends itself to a successive approximation of integers, we define the $k-$th level of approximation of integer x as

$$x_k = \text{sign}(x) \cdot (\sum_{i=0}^{k-1} b_i 2^i).$$

Since x_k may be equal to zero for some values of k while x may not, we attach the sign bit (for nonzero values of x) to its most significant bit b_i (see Table 2).

More generally, for any signed integer x, it can always be represented in the following way:

$$x = x_l + \text{sign}(x - x_l) \cdot (\sum_{i=0}^{l-1} b_i 2^i),$$

where $x_l = \lceil \frac{x}{2^l - 1} \rceil$ and the $k-$th level of approximation of x is defined as

$$x_k = x_l + \text{sign}(x_k - x_l) \cdot (\sum_{i=k}^{l-1} b_i 2^i).$$

Note that the sign bit is only necessary for $x_k \neq x_l$.

2.4 Bit-Plane Encoding

Given an image X and the result Y of its two-dimensional IWT, we decompose the transform image Y into

1. a truncated transform image $Z = (z_{i,j})$; and
2. l bit planes of information associated with l successive levels of approximation.

Due to different normalization factors for the transform coefficients in different subimages, a different binary decomposition is required for each subimage, for l levels of approximation sought. For example, consider just one level of decomposition of image X into the four subimages Y_{00_1}, Y_{01_1}, Y_{10_1}, and Y_{11_1}. To achieve the first level of approximation of Y, one must remove no bits from coefficients in Y_{00_1}, 1 bit from coefficients in Y_{01_1} and Y_{10_1}, and 2 bits from coefficients in Y_{11_1}.

This procedure can be generalized to any subimage Y_{00_i}, Y_{01_i}, Y_{10_i}, or Y_{11_i} (where i refers to the level of the IWT which produced the corresponding subimage). Let l be the number of levels of approximation sought:

- for Y_{00_i}:
  - if $i \geq l$, then no bits should be removed
  - if $i < l$, then $l - i$ bits should be removed
- for Y_{01_i} and Y_{10_i}:
  - if $i \geq l$, then no bits should be removed
  - if $i < l$, then $l - i + 1$ bits should be removed
- for Y_{11_i}:
  - if $i \geq l$, then no bits should be removed
  - if $i < l$, then $l - i + 2$ bits should be removed

Each bit plane taken from a subimage is assigned to the bit plane appropriate for a given level of approximation, along with the sign bits. If N is the total number of pixels, then the bit planes with a level of approximation $i > 1$ will contain about N bits (odd image widths and heights lead to a slightly lower number) and an undetermined number of sign bits. While at all approximation levels the number of sign bits is undetermined and very much image-dependent, their total number is at most N (i.e., the total number of pixels of transform image Y with nonzero values).

At each level of approximation, sign bits are encoded as they are, preceded by a header indicating how many sign bits are following. All bits other than the sign bits number N at each approximation level and are directly associated with pixels in the various subimages produced by the two-dimensional IWT. For a given level of approximation, all bits are grouped into 2×2 subblocks made of neighboring pixels in one of the subimages. When 2×2 subblocks cannot be formed due to odd widths or lengths of a subimage, then either a 2×1, or 1×2, or 1×1 subblock is formed. The bits of each subblock are padded together to form an integer varying between 0 and 15. Each bit plane is now made of a variable number of sign bits and a sequence of integers ranging from 0 to 15.

2.5 Entropy Coding

The transform image Y is then entropy coded in decreasing levels of approximation. The truncated image Z is entropy coded first. Then for each successive approximation level, the sign bits are encoded as they are, preceded by their number, while the other N bits (grouped into a sequence of integers ranging from 0 to 15) are entropy coded as well.

Any entropy-coding algorithm can be used here in both cases. In the following, we describe a simple run-length/Huffman encoding combination, which is a slight modification of an entropy coder designed by Eero P. Simoncelli and Edward H. Adelson at the Massachusetts Institute of Technology [1]. We will refer to this modified version as the EPIC entropy coder, and we will use it to evaluate compression performance.

2.5.1 Run-Length Encoding

The "run-length encoding" stage is used to code long runs of zeros as separate symbols, by uniquely decomposing the zero runs into a sequence of zero runs whose lengths are powers of two (2^i), using the run length's binary representation: a run of 9 zeros will be decomposed into a run of 8 zeros ($8 = 2^3$) and a run of 1 zero ($1 = 2^0$). The run-length coded image now contains additional symbols corresponding to run-lengths of 2^i, for $i = 1, \ldots$.

2.5.2 Huffman Encoding

After run-length encoding the subsignal, a histogram is computed for the run-length coded image, and a Huffman code is designed for that histogram. The Huffman code is encoded first using an algorithm used in [1] which we do not describe, as it is not part of the new technology we present here. Then the run-length encoded sequence is entropy coded with the Huffman code. Since all subsignals are encoded separately, each will require its own Huffman table to be encoded.

2.6 Algorithm Parameters

The following parameters can be selected by the user in the ERIC algorithm:

- nl_{low}: the number of times the simplified IWT is performed (obatined by dropping the high-frequency coefficients); it is recommended that 0 be chosen for low compression ratios and 1 for high compression ratios.

- nl_{lh}: the number of times the regular reversible IWT is subsequently performed; it is recommended that $nl_{low} + nl_{lh} \geq 4$.

- l: In lossless and lossy modes, l is the logarithm base two of the quantization factor ($Q = 2^l$).

  Thus, in lossless mode, l also signifies the number of bit planes to be encoded bit-plane-wise after quantization is applied. Bit-plane-wise endcoding provides the capability for progressive approximation of that portion of the coefficients remaining after quantization. Where N_{coeff} represents the number of bits per coefficient, the coarsest level of approximation is then $(N_{coeff} - l)$, which is achieved by a right shift l times, or (equivalently) by dividing each coefficient by 2^l. The dynamic range (number of bits) of the quantized coefficients—i.e., the coarsest level of approximation—must match the dynamic range of the entropy coder (which is currently 16 bits).

- compression mode: "lossless" will encode all bit planes corresponding to successive levels of approximation; "lossy" will simply quantize the transform image Y without any bit-plane encoding.

Another important choice is that of the entropy coder. In this report, we illustrate results based on an entropy coder designed by Eero P. Simoncelli and Edward H. Adelson of M.I.T., but it can be replaced by other entropy coding algorithms.

3 Decompression Algorithm

Given the relevant entropy-coded levels of approximation of the original image, they must each be entropy-decoded. Bit-plane decoding then takes place for the planes corresponding to the levels of approximation chosen. Then each integer transform coefficient is approximated according to the bits present in the bit planes and the sign bits.

Then the approximate integer transform image is inverse-transformed with the inverse IWT, one spatial resolution level at a time. The result is an approximation of the original image, the accuracy of which increases with the number of approximation levels being transmitted. If all levels are transmitted, the image is reconstructed exactly.

3.1 Entropy Decoding

Entropy decoding is exactly the reverse of entropy coding described earlier, decomposed into Huffman decoding the encoded bit stream, and run-length decoding the decoded bit stream.

3.1.1 Huffman Decoding

Huffman decoding consists first of decoding the Huffman table that was specifically designed for the specific data segment, then using it to decode the encoded bit stream.

3.1.2 Run-Length Decoding

After Huffman decoding, zero run-length symbols are replaced by their equivalent runs of zeros, and the transformed data stream has been reconstructed exactly.

3.2 Bit-Plane Decoding

Bit-plane decoding is exactly the inverse process of the bit plane encoding process described earlier: each entropy-decoded integer (whose range is between 0 and 15) is transformed into a 2×2 bit array (or 2×1, 1×2, or 1×1 at the boundary of a subimage).

3.3 Successive Approximation

Each integer x is reconstructed at the target level of approximation k

$$x_k = x_l + \text{sign}(x_k - x_l) \cdot (\sum_{i=k}^{l-1} b_i 2^i),$$

as described earlier.

3.4 Inverse Integer Wavelet Transform

The inverse two-dimensional IWT is simply the inverse procedure from that applied when computing the forward two-dimensional IWT. Each level of a two-dimensional inverse IWT is made of three one-dimensional inverse IWTs, and so we present both one- and two-dimensional IWTs.

3.4.1 One-Dimensional Transform

If the length N of transform signal Y is even (i.e., $N = 2K$), then the inverse integer transform sequence X is computed in the following two steps, even coefficients first

$$\begin{cases} x_{2i} = y_{2i} - \lfloor y_{2i+1}/2 \rfloor & \text{for } \; i = 0 \\ x_{2i} = y_{2i} - \lfloor (y_{2i-1} + y_{2i+1})/4 \rfloor & \text{for } \; i = 1, \ldots, K-1 \end{cases}$$

and odd coefficients second

$$\begin{cases} x_{2i+1} = y_{2i+1} + \lfloor (x_{2i} + x_{2i+2})/2 \rfloor & \text{for } \; i = 0, \ldots, K-2 \\ x_{2i+1} = y_{2i+1} + x_{2i} & \text{for } \; i = K-1 \end{cases}$$

If the length N of transform signal Y is odd (i.e., $N = 2K + 1$), then the inverse integer transform sequence X is computed in the following two steps, even coefficients first

$$\begin{cases} x_{2i} = y_{2i} - \lfloor y_{2i+1}/2 \rfloor & \text{for } \; i = 0 \\ x_{2i} = y_{2i} - \lfloor (y_{2i-1} + y_{2i+1})/4 \rfloor & \text{for } \; i = 1, \ldots, K-1 \\ x_{2i} = y_{2i} - \lfloor y_{2i-1}/2 \rfloor & \text{for } \; i = K \end{cases}$$

and odd coefficients second

$$x_{2i+1} = y_{2i+1} + \lfloor (x_{2i} + x_{2i+2})/2 \rfloor \text{ for } \; i = 0, \ldots, K-1$$

3.4.2 Two-Dimensional Transform

One level of a reverse IWT for images can be computed in two steps:

- two one-level inverse IWTs in the vertical dimension, producing two integer output subimages;
- one one-level inverse IWT in the horizontal dimension, producing one integer output image.

Further levels of the reverse IWT are again computed on the full subimage produced at the previous level and the three high-frequency subimages at the current level. This procedure is repeated as many times as required by the user but not more than the total number of levels of decomposition performed at compression (which would exactly reproduce the original image at full resolution).

3.5 Algorithm Parameters

The major parameter of the decompression algorithm is the approximation level (from 0 to b) at which the user wishes to reconstruct the signal, or the number of bytes from the encoded bit stream (equivalently, the compression ratio) the user wishes to use for reconstruction.

There are four decompression modes available with this algorithm:

1. "lossy with progressive spatial resolution": reconstruct the image using only the subimages of quantized transform image Z corresponding to the required level of spatial resolution;

2. "lossy": reconstruct the image using only the subimages of quantized transform image Z corresponding to the full level of spatial resolution (no bit plane information is used);

3. "lossy with rate control": recontruct the image with the use of a fixed number of bits from the compressed bit stream, including bits from the relevant bit planes; and

4. "lossy with quality control": reconstruct the image with the use of a limited number of bit planes.

4 Performance

This section presents the performance of the ERIC algorithm, both in terms of compression ratios achievable and in terms of compression/decompression timing. We compare ERIC to other standard lossless compression algorithms. We did not include comparisons to the CREW algorithm, which is also an algorithm based on an Integer Wavelet Transform, but for which no code is available for comparison purposes.

4.1 Lossless Compression/Decompression

Two modes of lossless compression/decompression can be considered depending on the application. The first consists of specifying zero levels of approximation ($l = 0$); the second, using $l = 8$, which allows for progressive encoding.

4.1.1 Lossless Performance with No Progressive Encoding

First, we compare the performance of ERIC as a purely lossless image compression algorithm. We compare it to that of two state-of-the-art algorithms: the RICE algorithm [4, 5], and the lossless mode of the JPEG standard algorithm [6], which is based on DPCM (Differential Pulse-Code Modulation) prediction.

Table 3: Lossless Compression Ratio Performance with No Progressive Encoding

Algorithm	Lena	Barb	Comet	Gaspra	Lena2	Mars
ERIC	1.77	1.48	2.05	6.89	1.60	1.61
JPEG (DPCM)	1.81	1.41	1.89	4.16	1.66	1.61
RICE	1.60	1.37	1.91	5.03	1.50	1.59

Table 4: Lossless Compression/Decompression Timing Performance (in seconds) with No Progressive Encoding

Algorithm	Lena	Barb	Comet
ERIC	1.1/0.8	1.2/0.9	1.0/0.7
JPEG (DPCM)	3.2/2.2	3.3/2.3	3.2/2.1
RICE	2.5/3.3	2.7/3.5	2.3/3.3

Compression Ratio

In Table 3, the lossless compression ratios of six sample images are given: four 512×512 images (Lena, Barbara, Comet, Gaspra) and two 256×256 images (Lena2, Mars). For Barbara and Comet, ERIC achieves a compression ratio 5 to 7 percent higher than that given by the better of the two algorithms; for Gaspra, the increase is 37 percent.

Timing

To illustrate the particularly low complexity of ERIC, in Table 4 we compare the lossless compression and decompression times obtained for the current version of ERIC on a Sun SPARCStation 10/41 for three of the sample images. The numbers given are the compression time (CT) followed by the decompression time (DT): CT/DT in seconds. A quick analysis of the results shows improvements in time by factors of 2 to 3 over the better of the JPEG lossless and RICE algorithms.

Another interesting observation is that the decompression time of this algorithm is roughly proportional to the number of reconstructed pixels at a given resolution level. This is illustrated in Figure 2.

4.1.2 Lossless Performance With Progressive Encoding

Next, we compare the performance of ERIC as a purely lossless image compression algorithm, using 5 levels of the IWT and 8 bit planes of approximation. We compare it to that of two state-of-the-art algorithms: the RICE algorithm [4, 5], and the lossless mode of the JPEG standard algorithm [6], which is based on DPCM prediction. It is important to note that neither of these two algorithms has acceptable progressive transmission or lossy compression capability.

Compression Ratio

Table 5: Lossless Compression Ratio Performance With Progressive Encoding

Algorithm	Lena	Barb	Comet	Gaspra	Lena2	Mars
ERIC	1.76	1.55	2.00	6.52	1.65	1.57
JPEG (DPCM)	1.81	1.41	1.89	4.16	1.66	1.61
RICE	1.60	1.37	1.91	5.03	1.50	1.59

Table 6: Lossless Compression/Decompression Timing Performance (in seconds) With Progressive Encoding

Algorithm	Lena	Barb	Comet
ERIC	1.8/2.0	1.9/2.1	1.8/2.0
JPEG (DPCM)	3.2/2.2	3.3/2.3	3.2/2.1
RICE	2.5/3.3	2.7/3.5	2.3/3.3

In Table 5, the lossless compression ratios of six sample images are given: four 512×512 images (Lena, Barbara, Comet, Gaspra) and two 256×256 images (Lena2, Mars). For Barbara and Comet, ERIC achieves a compression ratio 5 to 10 percent higher than that given by the better of the two algorithms; for Gaspra, the increase is 30 percent.

Timing

To illustrate the low complexity of ERIC, in Table 6 we compare the lossless compression and decompression times obtained for the current version of ERIC, using progressive encoding, on a Sun SPARCStation 10/41 for three of the sample images. The numbers given are the compression time (CT) followed by the decompression time (DT): CT/DT in seconds. A quick analysis of the results shows timing performance comparable to that of the other two lossless algorithms, thus confirming the low complexity of ERIC.

4.2 Lossy Decompression

The lossy algorithm chosen for comparison purposes is the JPEG algorithm [6] developed for the Imager for Mars Pathfinder, as its performance was better than other public-domain versions of JPEG. We will refer to this algorithm as the IMP-JPEG algorithm (version 3.3 is used for performance evaluation). Note that (unlike ERIC) the IMP-JPEG algorithm does not have an integrated lossless mode; i.e., they use a separate algorithm for lossless compression—the RICE algorithm.

4.2.1 Rate-Distortion Curves

The performance of lossy compression algorithms is most often expressed in operational rate-distortion curves, which are obtained by varying the compression parameter—e.g., quality factor, or target compression ratio. The two variables shown in these plots are the compression ratio (more conveniently displayed on a logarithmic scale) and the peak signal-to-noise ratio (PSNR), which is a logarithmic function of the mean square between the original and recontructed images. The operational rate-distortion curves given in Figure 3 are for a range of compression ratios achieved by IMP-JPEG (dotted curve), and a selection of ratios achieved by ERIC (solid curve). These performance curves were obtained for six images: four 512×512 images (Lena, Barbara, Comet, Gaspra) and two 256×256 images (Lena2, Mars). These curves show the superiority of ERIC over IMP-JPEG, except in the case of the Gaspra image, which features equal performance by IMP-JPEG and ERIC, and for which large blocks of pixel values are equal to zero.

Finally, it is important to note that (unlike ERIC) the rate-distortion points achieved by IMP-JPEG as displayed in Figure 3 are not achieved progressively.

References

[1] Edward H. Adelson and Eero P. Simoncelli, "Subband Image Coding with Three-Tap Pyramids," *Picture Coding Symposium,* 1990. Cambridge, Mass.

[2] Ahmad Zandi, Martin Boliek, Edward L. Schwartz, and Alexander Keith, "CREW Second Evaluation — ISO/IEC JTC1/SC29/WG1," 3 November 1995, *CRC-TR-9534*, ISO/IEC JTC1/SC29/WG1, N252.

[3] Robert O. Reynolds, Peter H. Smith, Devon G. Crowe, Mark Bigler, and Mike Pollard, "Design of a Stereo Multispectral CCD Camera for Mars Pathfinder," *SPIE Proceedings, Optomechanical and Precision Instrument Design*, Vol. 2542, Editor: Alson E. Hatheway; Alson E. Hatheway, Inc., 1995. pp. 197-206.

[4] Robert F. Rice, "Some Practical Universal Noiseless Coding Techniques," *JPL Publication 79-22*, 1979.

[5] Robert F. Rice, Pen-Shu Yeh, and Warner H. Miller, "Algorithms for a Very High-Speed Universal Noiseless Coding Module," *JPL Publication 91-1*, 1991.

[6] W. B. Pennebaker and J. L. Mitchell, *JPEG Still-Image Data Compression Standard,* Van Nostrand Reinhold, 1993.

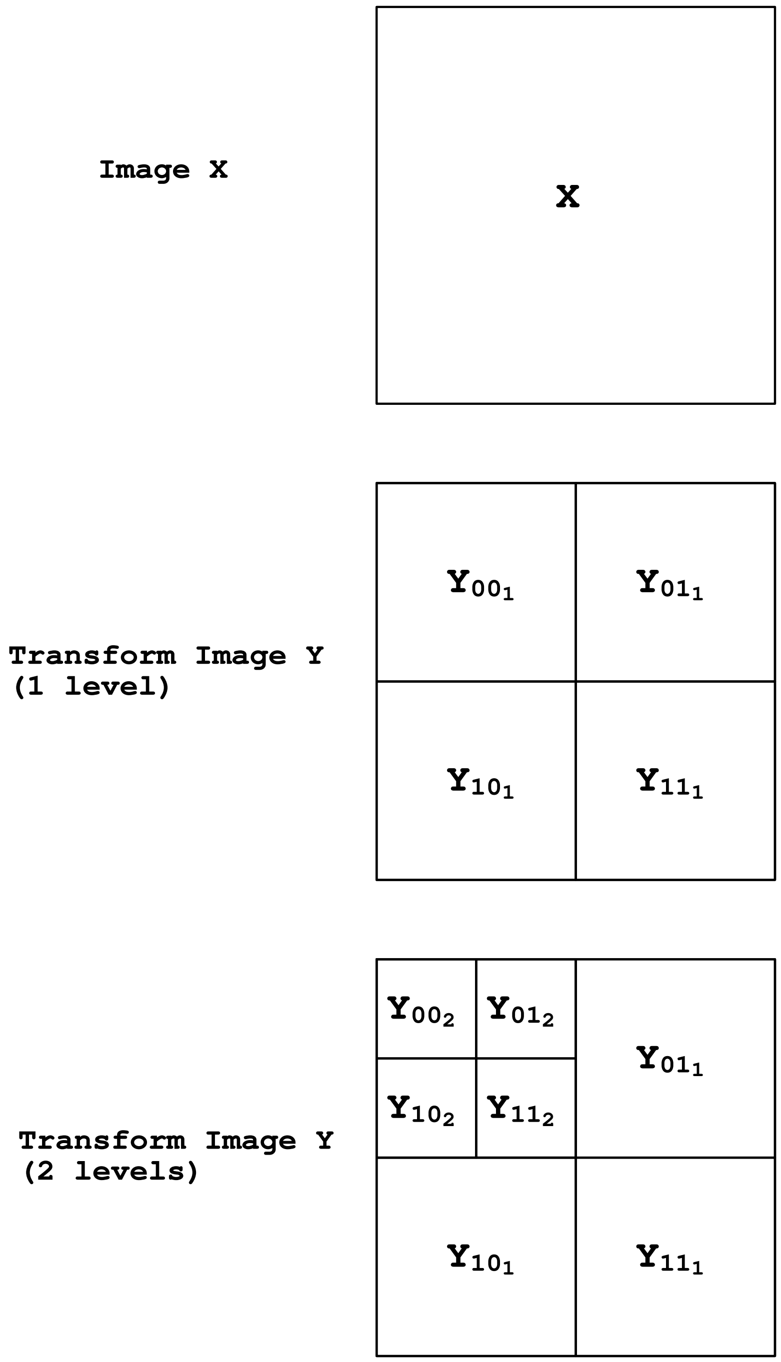

Figure 1: Two-Dimensional Wavelet Transform.

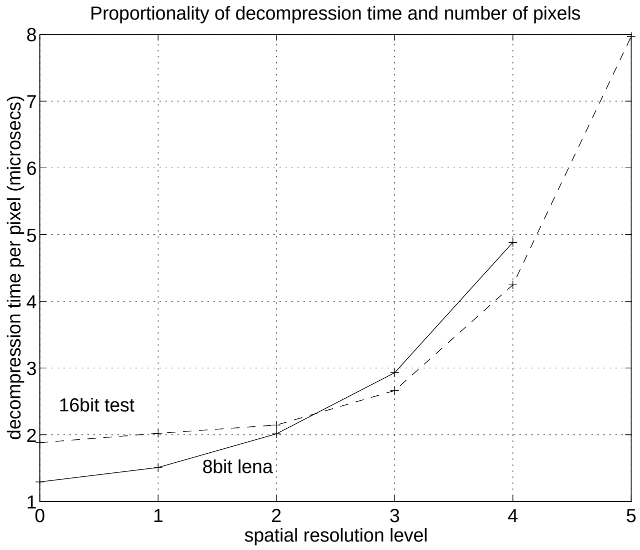

Figure 2: The decompression time per pixel reconstructed at a given resolution level becomes increasingly constant as the number of reconstructed pixels increases, depending on the image. It can reach about 1.3 μs on a Sun SPARCStation 10/41 under Solaris.

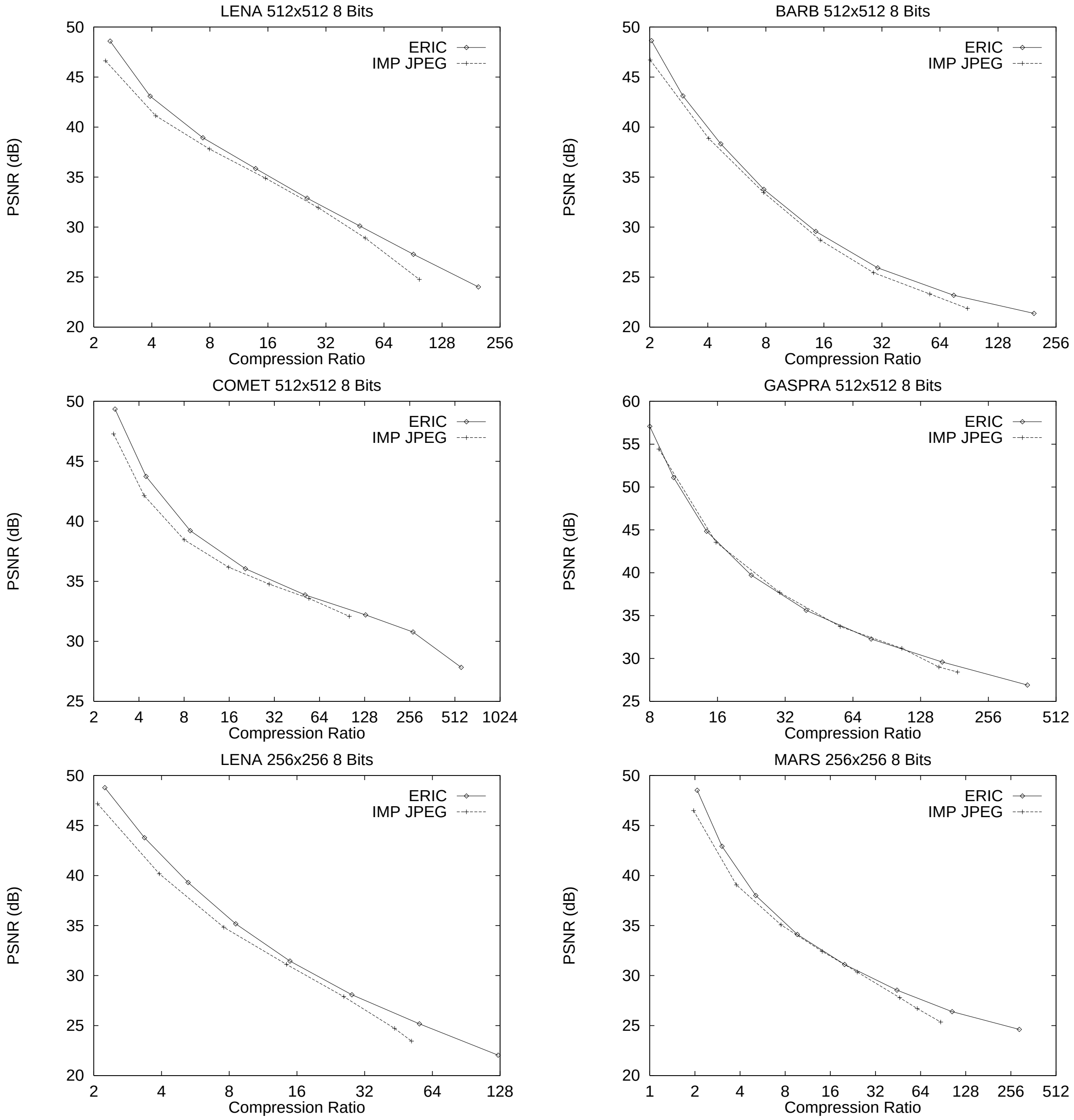

Figure 3: Operational Rate-Distortion Curves for ERIC and IMP-JPEG Algorithms.