

## Structured Trees for Lossless Coding of Quantized Wavelet Coefficients

Jiangtao Wen  
gwen@icsl.ucla.edu

Peyman Meshkat  
paymon@icsl.ucla.edu

John Villasenor  
villa@icsl.ucla.edu

Electrical Engineering Department  
University of California, Los Angeles

### Abstract

*We describe two low-complexity entropy coding structures for compression of quantized wavelet coefficients for image compression applications. The coders operate independently within each subband and represent zero runs and levels. The first coder uses a parameterized, structured coding tree in which the parameters are selected on an image- or subband-specific basis. The second coder uses a set of non-adaptive, non-structured coding trees. In both of the methods the coefficients are considered in raster order on an intrasubband basis. This leads to a simpler structure relative to zerotree algorithms, though in contrast with zerotree, the technique we use here does not produce an embedded bitstream. The main advantage lies in the ability of this approach to provide good performance (typically within 1 dB of the best PSNR results reported in the literature) at a cost which is significantly lower than other algorithms providing comparable PSNR results.*

### 1. Introduction

Recent work in wavelet image coding has resulted in quite a number of very advanced coding algorithms that give extremely good performance. Building on the signal processing foundations laid by researchers including Vetterli [1] and on the basic concepts of zerotree coding introduced by Shapiro [2], algorithms described by Said and Pearlman [3], Xiong et al. [4], and Joshi et al. [5] appear to be converging on performance limits of wavelet image coding. Substantially less attention has been given to algorithms in which the principal constraints are on complexity as opposed to performance. This does not imply that that algorithms listed above are overly complex; in fact, the algorithm of Said and Pearlman involves low arithmetic complexity. However, the question still arises as to how small the sacrifice in coding

performance can be made while reducing the complexity even further.

The principal result of the present paper is the introduction of entropy coding trees that are designed to match the pdfs of runs and levels in run-length coded wavelet subbands. In the interest of simplicity, we have chosen not to exploit the intersubband correlations in wavelet decompositions or the dependencies between neighboring nonzero wavelet coefficients within a single subband. Furthermore, arithmetic coding, which is used in many other wavelet compression algorithms and which involves a nontrivial complexity burden, is not employed here. Despite these simplifications, the coder described here performs in most cases within about 1 dB (and often less) in PSNR of the best results reported in the literature. Perhaps more significantly, it achieves the same PSNR as the non-arithmetically coded version of the Said and Pearlman algorithm, while requiring about half as much time to compute. The fact that we can obtain these PSNR results underscores our belief that while zerotree-like structures are of great value in creating embedded bitstreams, their value as aids to coding efficiency may not be as high as is often assumed. Since the algorithm utilizes a combination of run length coding and entropy coding trees, it handles quantized transform coefficients in a way that resembles the approaches used in image coding standards such as JPEG, MPEG, H.263, and the FBI wavelet compression standard, though with the difference that runs and levels are considered separately here. The separate handling of runs in levels represents another tradeoff that we have made in favor of lower complexity. The new aspects of this work are new tree structures we introduce for coding quantized wavelet coefficients.

Trees with structure have been used before in image coding. The most important example is the Golomb-Rice coder originally described in [6] and [7] that constitutes a key step in the LOCO-I [8] lossless image coding algorithm. Golomb-Rice codes are near-optimal for exponentially distributed sources. Another motivating factor in the present

paper is the stack-run coder described in [11], which utilizes a 4-ary symbol alphabet to efficiently map runs and levels into a symbol stream that is further compressed using adaptive arithmetic coding. The stack-run mapping can also be expressed using a structured tree, but one which is quite different from the tree representing a Golomb-Rice coder. Both the stack-run and Golomb-Rice coders can be expressed as special cases of a 2-parameter family of trees that we describe here. By proper parameter choice, an entropy coder can be specified that matches the pdfs of subband run lengths and coefficient levels very closely. Because the coder is described using only 2 parameters, it can be chosen on a subband-specific basis and the parameters can be transmitted with negligible overhead. It is also of interest to understand the performance obtainable when the constraint on structure is removed. To that end, we also consider here non-adaptive, non-structured coding trees and show that these trees also enable very good coding performance.

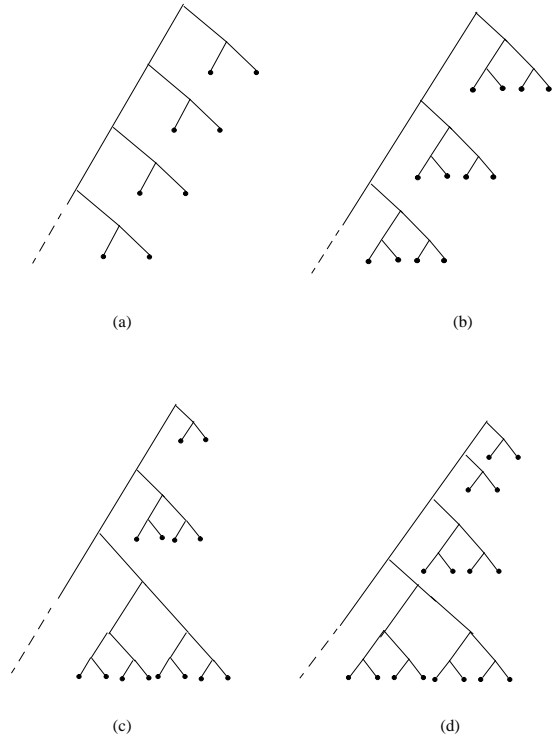
## 2. Description of the coders

The framework we use consists of a wavelet transform, uniform scalar quantization using a quantizer with a dead zone near the origin, run length coding and finally entropy coding of the run lengths and levels. Runs and levels are coded using entropy coding trees which assign codewords to run lengths and levels separately. If the quantized wavelet coefficients were i.i.d, then the pdf of the run lengths would be exponential, meaning that the Golomb-Rice code should be used for entropy coding of run lengths. However, as the normalized run length histograms plotted in Figure 2 show, in practice the distribution is not exponential. As will be discussed below, it is possible to use alternative coding trees that still have structure, and that match the statistics of runs and levels encountered in real images much more closely.

### 2.1 A structured, parameterized family of trees

Structured trees can be easily parameterized and can be used to match a wide range of input statistics. They are especially desirable for information sources with a large (or even infinite) alphabet, for which designing and using optimally matched trees in the Huffman sense involves high costs. Structured code trees have long been used in lossless entropy coding. The most widely known example is the Golomb-Rice coder for exponentially distributed sources of various decay rates. The ideas behind Golomb-Rice codes were originally proposed in [6] and [7], and have recently applied for coding of prediction errors in lossless image coding applications [8]. Golomb-Rice codes are nearly optimal for coding of exponentially distributed non-negative integers, and describe an integer  $n$  in terms of a quotient and a remainder. For simplicity, the divisor is often chosen

to be a power of 2,  $2^k$ , and is parameterized by  $k$ . The quotient can of course be arbitrarily large and is expressed using a unary representation; the remainder is bounded by the range  $[0, 2^k - 1]$  and is expressed in binary form using  $k$  bits. For example, for a Golomb-Rice code with  $k = 2$  the number 9 could be represented as 00101. The first two zeros, terminated by the first one, identify the quotient of  $10/2^2$  as having value 2; the 01 is a 2-bit binary expression of the remainder.

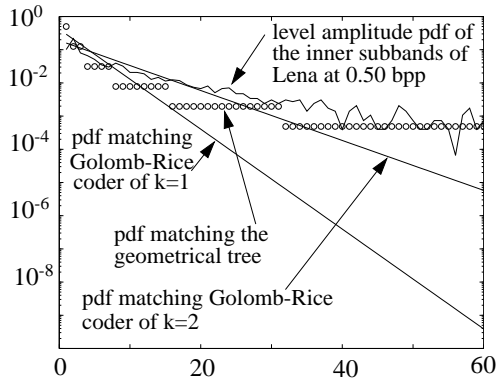


(a) Golomb-Rice tree,  $k = 1$  (b) Golomb-Rice tree,  $k = 2$   
(c) Geometrical tree (d) Hybrid tree,  $k = 1, l = 1$

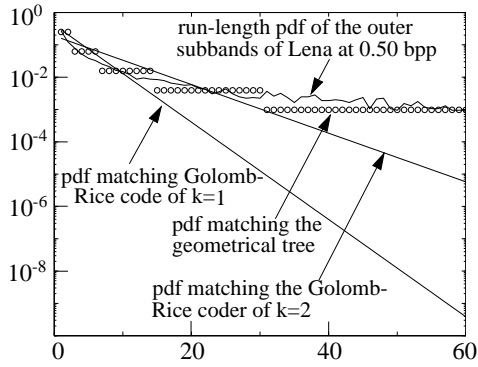
**Figure 1. Coding Trees**

Figures 1 (a) and (b) show the trees corresponding to Golomb-Rice codes with parameters  $k = 1$  and  $k = 2$  respectively. Note that at each level in the Golomb-Rice tree there are  $2^k$  code words. The pdfs to which these trees are matched are shown in Figure 2, and as one would expect for a coder matched to exponential sources, fall off linearly when plotted on a semilog scale.

In another publication [11], we have described a coding algorithm called stack-run coding that performs well for quantized wavelet coefficients. Although the stack-run coder was not discussed in [11] in terms of a prefix code, it utilizes a mapping which is equivalent to the tree shown in Figure 1 (c), in which the number of codewords of a given length grows geometrically with codeword length.



(a) Level pdf



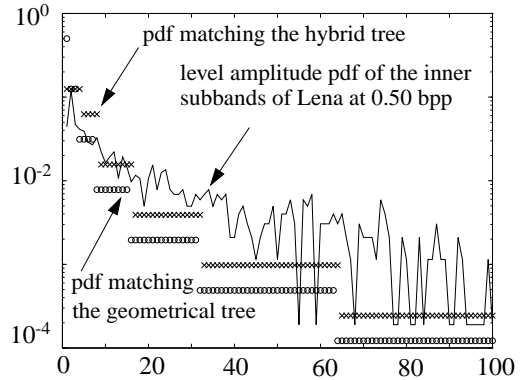
(b) Run pdf

**Figure 2. Comparison of the histogram from wavelet coefficients of the Lena image with the pdf matched to Golomb-Rice and geometrical trees. In this and all other figures the 9/7 filters with a 5-level octave band decomposition and uniform quantization were used.**

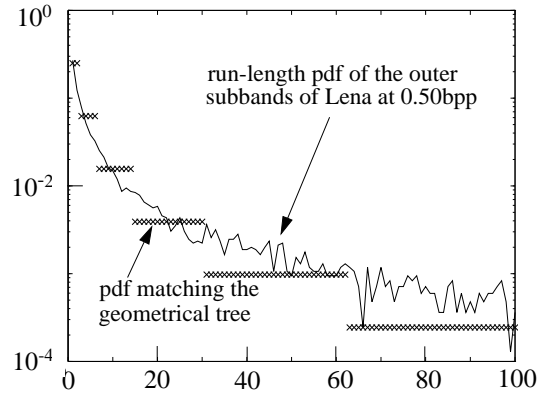
For this reason, we refer to this tree as an “geometrical” tree. The number of codewords  $N_i$  of length  $i$  in the geometrical tree satisfies

$$N_i = \begin{cases} 2^{i/2} & i \text{ even} \\ 0 & i \text{ odd} \end{cases} \quad (1)$$

The pdf to which the geometrical tree is matched is plotted in Figure 2, along with the pdfs for two parameter choices of the Golomb-Rice tree. The figures also contain a set of normalized histograms from the quantized wavelet coefficients from the Lena image obtained using a 5-level dyadic decomposition using the 9/7 filters. The quantization step size was uniform across all subbands and was chosen to give a coding rate of .5 bpp when the algorithm of structured tree coding described in this paper is used. Figure 2 (a) contains



(a) Level pdf



(b) Run pdf

**Figure 3. Comparison of the normalized histogram with the pdf for Hybrid Tree Coding.**

the histogram for the levels from the 7 innermost subbands, and Figure 2 (b) contains the histogram from the run-lengths in the outer 9 subbands. We made this choice because the coefficients in the inner subbands tend to be non-zero, while there are many consecutive zeros in the outer subbands. At high indices (e.g. large levels or long runs), it is clear that the geometrical tree is much more closely matched to the actual distributions. For lowest indices, though it is not evident in the scaling used in this figure, the Golomb-Rice tree actually gives a better match to the data for the levels (Figure 2 (a)). By contrast, the geometrical tree matches the runs well across all indices. Although the number of indices for the level histogram for which the Golomb-Rice tree is better is very small, this is very important from a coding efficiency standpoint since these indices occur with the highest probability. This suggests coding the levels using a hybrid tree that has the attributes of the Golomb-Rice coder

for small indices, and the geometrical tree for larger indices. An example of a hybrid tree as shown in Figure 1 (d). A hybrid tree contains the first  $l$  levels of a Golomb-Rice tree with divisor  $2^k$ , and thereafter is a tree with an geometrical structure. As a function of the parameters  $k$  and  $l$ , the number of codewords  $N_i$  of length  $i$  satisfies

$$N_i = \begin{cases} 0 & \text{if } i < k + 1 \\ 2^k & \text{if } k + 1 \leq i \leq k + l \\ 2^{(k+i-l-1)/2} & \text{if } i \geq k + 1 + l \text{ and } k + i - l \text{ is odd} \\ 0 & \text{if } i \geq k + 1 + l \text{ and } k + i - l \text{ is even} \end{cases} \quad (2)$$

Figure 3(a) shows a plot of the pdf of the hybrid tree for parameters  $(k, l)=(1, 1)$  and of the same run and level data from the Lena image as used in Figure 2. The vertical scale has been enlarged in Figure 3 relative to the plots in Figure 2. It is clear from Figure 3 that the hybrid tree matches the data better than the geometrical tree. This is borne out by entropy calculations. The first order entropy of the subband level data shown in Figure 3(a) is about 4.1 bits/sample (though this is not generally achivable because the entropy calculation is performed based on only one realization). The best Golomb-Rice tree requires over 10 bits/sample to code these data. The geometrical tree uses 6.7 bits/sample, and the hybrid tree shown in the figure with  $(k, l) = (3, 1)$  uses 6.3 bits/sample.

The parameterized tree structure enables very simple encoding and decoding. Let  $L(j)$  be the  $j$ th shortest code-length. For example, for a Golomb-Rice coder with parameter  $k$ ,  $L(j) = k + j$ . For the geometrical tree  $L(j) = 2j$ . For the hybrid tree in Figure 2 (d)  $L(j) = 2, 3, 5,$  and  $7$  for  $j = 1$  to  $4$ . Furthermore, when a codeword is of length  $L(j)$ , we call it a codeword of layer  $j$ . Using this notation, we can find that for any parameter choice, codeword lengths satisfy

$$N_{L(j)} \cdot 2^{-L(j)} = 2^{-j} \quad (3)$$

If we further denote

$$m_j = \log_2 N_{L(j)} \quad (4)$$

then

$$L(j) = j + m_j \quad (5)$$

which means that we can divide each codeword of length  $L(j)$  into two parts, a  $j$ -bit part and a  $m_j$ -bit part. If we organize the tree as shown in Figure 1 (d), then the first  $j$  bits of any codewords of layer  $j$  will be the unary expression of  $j - 1$ . This part of the codeword is used to communicate information on the total length of the codeword (which can be uniquely determined by the layer number  $j$ ), so we call it “layer prefix”. Specifying any particular one of the  $N_{L(j)} = 2^{m_j}$  codewords of layer  $j$  is the function of the next  $m_j$  bits of the codeword, and thus we name this part the “intra layer index”.

When coding an information source of positive integers with monotonically decreasing pdf with codes of the above described structure, we can decide the appropriate layer for its codeword by comparing the input integer with the integer corresponding to the first codeword of each layer, and the intra layer index is the binary expression of the input integer minus that corresponding to the first codeword of that layer. The structure of the codes makes it easy for the decoder to separate “layer prefix” and “intra layer index” parts of the code, and the decoded integer is simply the sum of the “intra layer index” and the integer corresponding to the first codeword of the layer indicated by the “layer prefix”.

Having identified a family of trees that is well-suited to quantized subband data, the problem still remains of finding an efficient way to perform the parameter choice on an image-specific basis. By examining the statistics of runs and levels for a range of images and coding rates, we have found that for run lengths, the geometrical tree is nearly optimal. For levels, the method we use involves a low-complexity estimation of the number of bits that will be required to code the data using each of several candidate parameter choices. This estimation is easy to perform for a structured tree. To use the above codes and algorithms to image coding, we still have to solve two more problems: coding the sign of levels and distinguishing runs and levels. To solve the first problem, we can design the codebook as shown in Table 1 so that the last bit of every codeword is the sign bit.

Layers	Codewords	Corresponding Level	Corresponding Run
1 $m_1=2$	1 0	1	1
	1 1	-1	2
2 $m_2=2$	01 0	2	3
	01 1	-2	4
3 $m_3=4$	001 00	3	5
	001 01	-3	6
	001 10	4	7
	001 11	-4	8

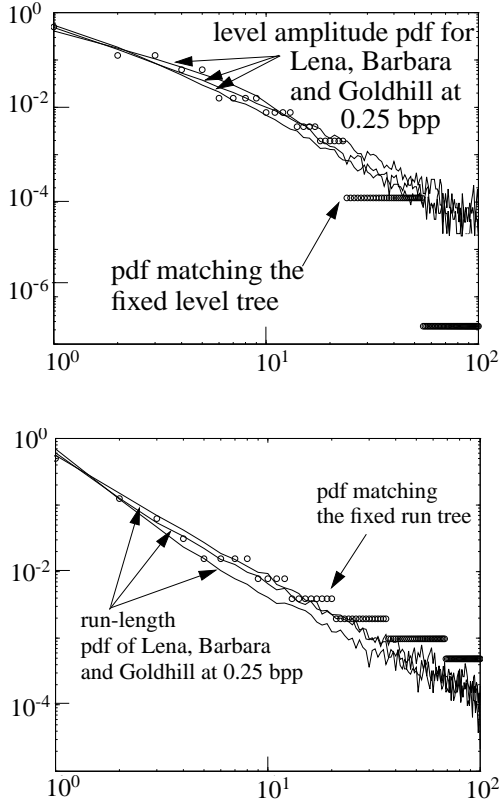
Table 1: Sample Codebook for the Hybrid Tree of  $k=1, l=1$

To solve the second problem, we note that there is always a level after a run, but a level can be followed by either a level or a run. This can be handled using a one-bit flag after the codeword for each level that tells the decoder whether the next codeword represents a run or level. This is equivalent to coding zero run-lengths with one bit and adding a bit to the representations of all other run-lengths.

## 2.2 Fixed entropy coding trees

It is also of interest to know what performance can be obtained if one removes the constraint that the trees used for coding the runs and the levels have structure, using instead

a single tree designed to match the cumulative statistics of runs or levels across all subbands.



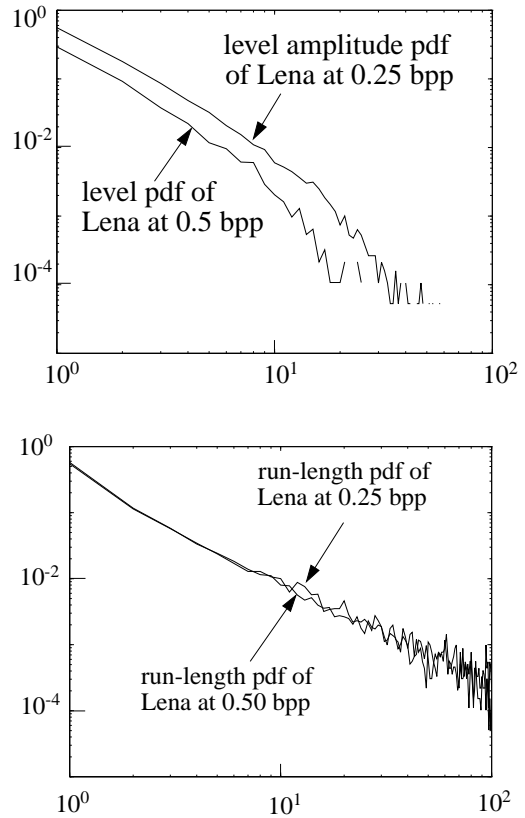
(x axis represents Run+1 to avoid zero on log scale).

**Figure 4. Top figure is Fig. 4a; bottom is Fig. 4b. Statistics of run lengths and levels for 3 different test images. (Cumulative statistics across all but four inner subbands)**

Although the statistics of runs and levels differ greatly across subbands, the cumulative statistics are very similar across different images and different quantization step sizes. This is illustrated in Figures 4 and 5.

Figure 4 (a) and (b) show (solid lines in the figures) the pdf of run lengths and absolute value of levels for Lena, Barbara and Goldhill when quantization consistent with a coding rate of approximately .25 bpp is performed.

The decomposition used was the same as before, a 5-level dyadic decomposition using the 9/7 filter. As the figures show, with the exception of Figure 5 (a), the histograms within each of these images differ only slightly. It is interesting to note that the cumulative statistics of runs as plotted using the log-log scale of Figure 4 (b) appear nearly linear, corresponding to a pdf of the form  $a_1(n+1)^{-b_1}$  with  $a_1$  and  $b_2$  positive constants. This contrasts with the exponential



(x axis represents Run+1 to avoid zero on log scale).

**Figure 5. Top figure is Fig. 5a; bottom is Fig. 5b. Pdf of run lengths and absolute value of levels for two different quantization step sizes which correspond to 0.25 bpp and 0.5 bpp respectively after quantization. (Cumulative statistics across all but 4 inner subbands)**

model  $a_2 b_2^{-n}$  often used for run lengths. This arises because the use of cumulative statistics corresponds to mixing several random variables with different decay rates.

Figure 5 shows the pdf of run lengths and absolute value of levels for the Lena image for two different stepsize corresponding to coding rates of .25 bpp and 0.5 bpp. Observing the similarities in the slopes of the curves suggests the possibility of designing fixed entropy coding trees, one optimized for levels and the other for runs. The four inner subbands are not run length coded and are entropy coded using a third fixed entropy coding tree. We designed three such trees by simply choosing codeword lengths that satisfy (to the precision allowed by integer rounding)  $l_i = \log_2 p_i$  where  $l_i$  is the length of the  $i$ th codeword and  $p_i$  is the probability of the corresponding symbol derived from the experimental results. The pdfs for which the trees are optimum are shown

(using small circles) in Figures 4 (a) and (b).

In Figure 4 there is a mismatch between the designed tree and the pdf of the real data. However the probability of the symbols corresponding to these values is less than what has obtained from the experimental data because the number of samples from which these values have been obtained is not statistically sufficient. The actual probability of symbols in this area is less than what has estimated from the samples of the test image.

### 3. Experimental results

We have applied the two methods described in this paper to several test images for different bit rates. As mentioned earlier, in the interest of reducing the implementation complexity we have not used arithmetic coding. We have used a fixed quantization step-size across all subbands. Although this approach is not optimal, it makes the implementation simpler.

Table 2 shows the PSNR values obtained using the coding trees described in this paper. We also include results for several other well known algorithms in the table. In all experiments we used a simple 5-level octave band decomposition using the 9/7 filters. In the experiments for structured tree compression, we used the geometrical tree ( $k = 1, l = 0$ ) for the compression of run-lengths, while the parameters of the code tree for levels were determined adaptively using the approach described in section 2. Inner and outer subband levels were treated separately. The cost for communicating the parameter choice is only a few bits per image, and therefore involves essentially no bandwidth overhead at practical coding rates. In the fixed tree method, run-lengths and levels were coded separately. The four inner subbands in 5 level decomposition (16 subbands) were not run length coded.

As the table shows the results of the two methods described here and of the Said and Pearlman algorithm without arithmetic coding are essentially the same. Our results are about 0.5 dB below the Said and Pearlman results with arithmetic coding and between 0.7 to 1.0 dB below the results given by Xiong [4] et al. Although neither our code nor the Said and Pearlman code is optimized for speed, in the timing experiments we have performed to date with the hybrid entropy coding tree, we find that the coder described here runs approximately twice as fast as the Said and Pearlman coder without arithmetic coding.

### 4. Conclusions

We have introduced very simple image coders which involve a wavelet transform, run length coding of quantized wavelet transforms, and finally symbol by symbol entropy

Algorithm	Goldhill		Barbara		Lena	
	0.5bpp	0.25bpp	0.5bpp	0.25bpp	0.5bpp	0.25bpp
Structured Tree	32.65 dB	30.14 dB	30.75 dB	27.04 dB	36.60 dB	33.45 dB
Fixed Tree	32.68 dB	30.11 dB	30.73 dB	27.08 dB	36.70 dB	33.62 dB
Said and Pearlman, w/o arith. coder [3]	32.68 dB	30.18 dB	30.70 dB	27.00 dB	36.82 dB	33.68 dB
Said and Pearlman, w/ arith. coder [3]	33.12 dB	30.56 dB	31.25 dB	27.40 dB	37.21 dB	34.11 dB
Shapiro [2]	n/a	n/a	30.53 dB	26.77 dB	36.28 dB	33.17 dB
Xiong, et al [4]	n/a	n/a	32.25 dB	27.96 dB	37.42 dB	34.35 dB
Stack-Run [11]	32.92 dB	30.30 dB	30.94 dB	27.32 dB	36.67 dB	33.58 dB

Table 2: Summary of Compression Results  
(All results except Xiong's and Shapiro's are obtained using the 9/7 filter, with a 5 level dyadic decomposition)

coding of run lengths and levels separately. In the first method we have introduced a structured entropy coding tree which is characterized by a single pair of parameters which are adjusted to the statistics of a given image and in the second method we have used fixed entropy coding trees for run lengths and levels. While the PSNR results are quite close to results derived from Said and Pearlman method, the computational complexity appears to be substantially lower.

### 5. Acknowledgment

This work was supported by Texas Instruments and DARPA/ITO under contract DABT63-95-C-0100.

### References

- [1] M. Vetterli and C. Herley, "Wavelets and filter banks: Theory and design," *IEEE Trans. on Signal Proc.*, vol. 40, pp. 2207-2232, 1992.
- [2] J. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients", *IEEE Tran. on Signal Processing*, vol. 41, no. 12, pp. 3445-3462, December, 1993.
- [3] A. Said, W.A. Pearlman A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, June 1996.
- [4] Z. Xiong, K. Ramchandran and M. T. Orchard, "Wavelet packets image coding using space-frequency quantization," submitted to *IEEE Trans. Image Processing*, January 1996.
- [5] R.L. Joshi, H. Jafarkhani, J.H. Kasner, T.R. Fischer, N. Farvardin, M.W. Marcellin, and R.H. Bamberger,

“Comparison of different methods of classification in subband coding of images,” submitted to *IEEE Trans. Image Proc.*, 1995.

- [6] S.W. Golomb, “Run-length encodings,” *IEEE Trans. Inf. Theory*, vol. IT-12, pp. 399-401, July 1966.
- [7] R.F. Rice, “Some practical universal noiseless coding techniques,” Tech. Rep. JPL-79-22, Jet Propulsion Laboratory, Pasadena, CA, March 1979.
- [8] Weinberger, M.J.; Rissanen, J.J.; Arps, R.B. Applications of universal context modeling to lossless compression of gray-scale images. *IEEE Transactions on Image Processing*, April 1996, vol.5, (no.4):575-86.
- [9] A. Zandi, J.D.Allen, E.L.Schwartz, M.Boliek, “CREW: Compression with reversible embedded wavelets,” *Proc. of the 1995 IEEE Data Compression Conference*, Snowbird, UT, pp. 212-221, March, 1995.
- [10] X.Wu, N.Memon, “Context-based, adaptive, lossless image coding”, Pre-print 1996. Accepted for publication in *IEEE Trans. Communications*.
- [11] M.J. Tsai, J. Villasenor, and F. Chen, "Stack-run image coding," *IEEE Transactions on Circuits and Systems for Video technology*, vol. 6, pp. 519-521, October 1996.
- [12] T. Hopper, C. Brislawn, and J. Bradley, "WSQ Gray-Scale Fingerprint Image Compression Specification, Federal Bureau of Investigation, Washington D.C., February, 1993.