Quadtree Classification and TCQ Image Coding*

Brian A. Banister and Thomas R. Fischer

School of Electrical Engineering and Computer Science

Washington State University

Pullman, WA 99164-2752

e-mail: bbaniste, fischer@eecs.wsu.edu

December 16, 1998

Abstract

The SPIHT wavelet image coding algorithm can be interpreted as implicitly using classification in its bit-plane coding procedure. The source distribution induced by this classification is studied and rate-distortion performance is evaluated. A quadtree sorting procedure, similar to SPIHT, is used to explicitly form classes of wavelet coefficients. The classes are encoded using arithmetic and trellis coded quantization. The resulting encoding algorithm offers consistent improvement over SPIHT performance due to the granular gain of the trellis code.

Note: Portions of sections II. C. and II. D. are included in B. A. Banister and T. R. Fischer, "Quantization performance in SPIHT and related wavelet image compression algorithms," submitted to *IEEE Signal Processing Letters*, July 30, 1998. This material is included to aid the reviewing process.

*This work was supported in part by the National Science Foundation under Grant CCR-9627815.

I. Introduction

The set partitioning in hierarchical trees (SPIHT) image coding algorithm [2] is fast and efficient, generates a fully embedded bit stream, and is highly competitive with other image coding algorithms. This paper shows that SPIHT can be viewed as a combination of bit-plane-based classification, implicit uniform scalar quantization, and entropy coding.[1] Like the embedded zerotree wavelet (EZW) coding algorithm [10], SPIHT uses a wavelet subband decomposition and imposes a quadtree structure across the subbands, seeking to exploit interband dependencies. The SPIHT algorithm alternates between sorting and refinement passes to encode each bit-plane. By viewing the SPIHT sorting process as a form of classification, this paper studies the underlying source distribution induced by the classification and the corresponding rate-distortion characteristics. A new coding algorithm is then described, based on using a modified version of the quadtree sorting procedure in SPIHT to explicitly form classes of wavelet coefficients, and trellis coded quantization (TCQ) [14], [6] to encode each class. The new encoding algorithm provides improved encoding performance over SPIHT due to the granular gain of the TCQ.

[1] The SPIHT algorithm does not require the use of entropy coding. Using entropy coding provides slightly better compression performance, and is the version of SPIHT investigated here.

II. SPIHT Coding

II. A. Implicit Classification

The SPIHT algorithm [2] imposes a hierarchical quad-tree data structure on a wavelet transformed image. The terms wavelet coefficient, subband coefficient, and subband pixel will be used interchangeably. Figure 1 indicates the parent-offspring relationship across the subbands. The set of root node and corresponding descendents is referred to as a spatial orientation tree (SOT). Three lists are used in encoding: the list of significant pixels (LSP); the list of insignificant pixels (LIP); and the list of insignificant sets (LIS). The LSP is initialized to be empty, the LIP is initialized with the elements of the lowest frequency subband, and the LIS is initialized with the root of each SOT.

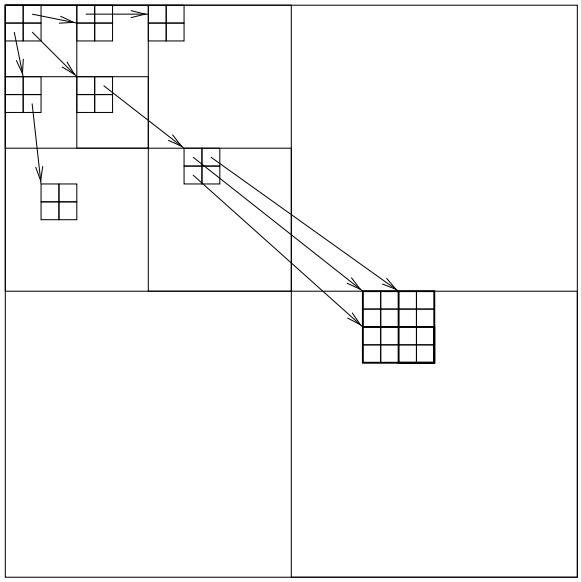

Figure 1: Quad-Tree organization of subband pixels in SPIHT algorithm.

After initialization, a threshold is chosen as $T(0) = 2^{n_0}$, where n_0 is selected such that the largest pixel magnitude, say c, satisfies $2^{n_0} \leq c < 2^{n_0+1}$. The encoding is progressive in pixel magnitude, using a sequence of thresholds

$$T(n) = 2^{n_0-n}, n = 0, 1, 2, \cdots. \tag{1}$$

Since the thresholds are a power of two, the encoding method can be thought of as "bit-plane" encoding of the wavelet coefficients. At stage n, all pixels with magnitudes satisfying $T(n) \leq |x| < 2T(n)$ are identified as "significant," and their position and sign bit encoded. This process is called a sorting pass. Then, every pixel with magnitude at least $2T(n)$ is "refined" by encoding the nth most significant bit. This is called a refinement pass. The encoding of significant pixel position, and the scanning of pixels for refinement, is efficiently accomplished using the LSP, LIP, and LIS.

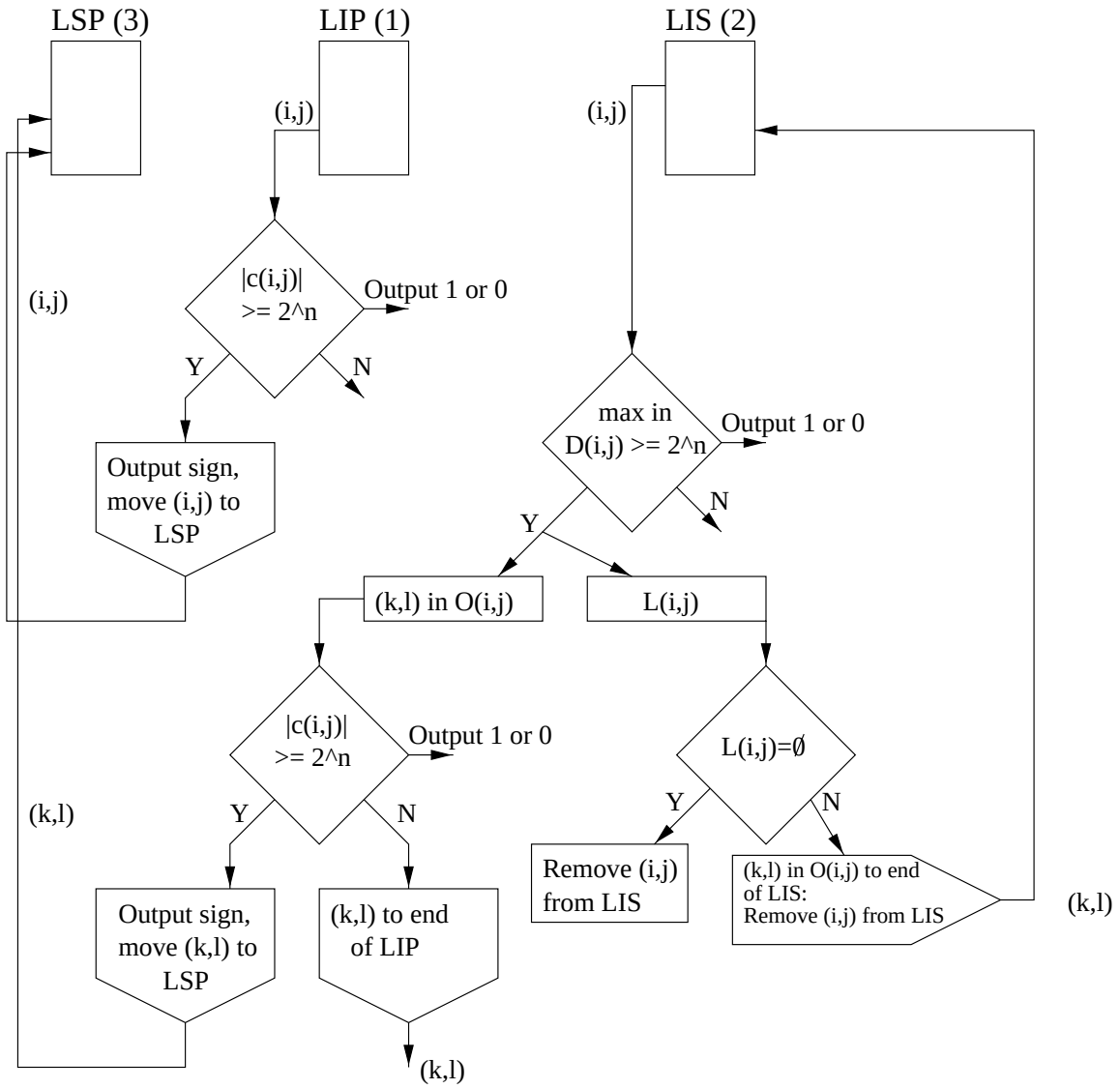

Figure 2: Flowchart of SPIHT encoding [13]

Let (i,j) denote the $(row, column)$ position of a coefficient in the transformed image, or node in a SOT. The set of immediate descendents (offspring) is denoted $O(i,j)$, the set of all descendents $D(i,j)$, and the set of all descendents not immediate offspring $L(i,j) = D(i,j) - O(i,j)$. Let $|(i,j)|$ denote the magnitude of the coefficient (i,j), and $|D(i,j)|$ the largest magnitude of any coefficient in $D(i,j)$. The encoding process in SPIHT is shown as a flow chart in Figure 2. The process proceeds as follows.

For $n = 0, 1, 2, \ldots$

1. Test each coefficient, c_k, in the LIP for significance, outputting one bit per coefficient. If the magnitude of any coefficient is significant ($|c_k| \geq 2^{n_\circ - n}$), then the coefficient is moved to the LSP, and the sign bit is encoded.

2. For each node in the LIS, if any descendent is found significant ($|D(i,j)| \geq 2^{n_\circ - n}$), the children are moved to the LSP and LIP as appropriate. The node is split into four descendent nodes which are placed at the end of the LIS.

The implicit classes in the SPIHT algorithm can be defined recursively using the sequence of thresholds in (1). Let $B(i,j)$ denote the set of coefficients that are the siblings (immediate offspring of the parent) of (i,j) [2]. Finally, define S_{-1} as the set of all wavelet coefficients, and LFS the set of wavelet coefficients in the lowest frequency subband. The SPIHT classes C_n, $n = 0, 1, \cdots$, are defined as

$$C_0 = \text{LFS} \bigcup \left\{ (k,l) \in \left(B(i,j) \bigcup D_0^{-1}(i,j) \right) : (i,j) \in S_0 \,\text{and}\, T(0) \leq |(i,j)| < 2T(0) \right\},$$

$$C_n = \left\{ (k,l) \in \left(B(i,j) \bigcup D_n^{-1}(i,j) \right) : (i,j) \in S_n \,\text{and}\, T(n) \leq |(i,j)| < 2T(n) \right\},\ n > 0,$$

where $S_n = S_{n-1} - C_{n-1}$, $C_{-1} = \{\emptyset\}$, and $D_n^{-1}(i,j)$ is the set of all ancestors in the SOT to (i,j) not in $\bigcup_{l=-1}^{n-1} C_l$. That is, the class C_n contains all coefficients, and their ancestors, of any block with largest magnitude in $[T(n), 2T(n))$ and not in classes $C_{n-1}, \cdots, C_0$. However, the class C_0 is initialized to also contain all LFS coefficients.

[2] For an image of size $2^N \times 2^N$, $B(i,j)$ is a 2×2 block of coefficients including (i,j). For other image sizes the SOT quadtree is modified to use additional block sizes.

II. B. Bit-plane coding

The SPIHT algorithm generates an embedded bit stream, using a method of bit-plane coding based on uniform scalar quantization. Let the quantization levels be represented in a sign-magnitude format as

$$x = (b_1, b_2, ..., b_k; b_s),$$

where $b_1, ..., b_k$ are the least to most significant bits of the magnitude, and b_s is the sign bit. Suppose we wish to losslessly encode all x in a class. For a memoryless source the ideal codeword length is $-\log_2 P(x)$. Using the sign-magnitude representation above, and using Bayes rule for conditional probabilities,

$$P(x) = P(b_s|b_1, ..., b_k) \cdot P(b_1|b_2, ..., b_k) \cdots P(b_{k-1}|b_k) \cdot P(b_k).$$

The ideal codeword length is then

$$-\left[\log_2 P(b_k) + \sum_{i=1}^{k-1} \log_2 P(b_i|b_{i+1}, ..., b_k) + \log_2 P(b_s|b_1, ..., b_k)\right].$$

Define the contexts

$$\mathcal{K}_i = \{b_{i+1}, ..., b_k\}, \quad \text{for } i = 1, ..., k-1,$$

$$\mathcal{K}_k = \emptyset,$$

and

$$\mathcal{K}_s = \{b_1, ..., b_k\}.$$

To progressively encode a block of source samples, one straightforward algorithm is to encode b_i in the context of $\mathcal{K}_i$ for every source sample, for $i = k, ..., 1, s$. Since an arithmetic code can achieve a rate approaching the entropy of the source, if arithmetic codes are used to encode each b_i in the context of $\mathcal{K}_i$, and assuming a large number of source samples, then this simple progressive encoding algorithm can achieve an average codeword length approaching the entropy $H(X)$. It is evident, then, that for a memoryless source, a progressive encoding

algorithm *is able to* provide an average codeword length approaching the source entropy *for the lossless coding of the quantization indices.*

Now let's consider the quantization implicit in embedded coding, and the implications of truncating the bit stream. Suppose that the encoding algorithm puts in the bit stream the encoded magnitude bits $b_k, b_{k-1}, ...$, with the encoded sign bit, b_s placed immediately following the first non-zero magnitude bit. Consider the quantization that has occurred after encoding (or decoding) of any particular bit b_i. Suppose that the first non-zero bit is b_k. The sign bit follows immediately, and the resulting quantization regions are shown in Figure 3. Note that this is a three-level scalar quantizer with step size 2^k. Note further that for the entropy-coded scalar quantization of a zero-mean, symmetric, unimodal source, this is exactly the form of the quantizer that would be used for a range of small (less than 1 bit/sample) encoding rates [5]. When the next bit is encoded (bit b_{k-1}) there are seven quantization regions, as shown in Figure 4. For source samples with $b_k = 1$, the bit b_{k-1} provides refinement of the outer quantization regions shown in Figure 3. For source samples whose MSB is zero, again a 3-level quantizer is used, (assuming the corresponding sign bit follows immediately if $b_{k-1} = 1$), with a step size of 2^{k-1}. With each successive bit transmitted, the process continues, yielding a sequence of quantizers, with 3, 7, 15, 31, ... quantization levels. Comparison of this sequence of quantizers with those entropy-constrained scaler quantizers (ECSQ) of [5], reveals that the quantizers implicit in the progressive bit-plane coding are (except perhaps for the width of the deadzone) precisely those quantizers that can be used in ECSQ *for selected encoding rates.* Assuming an arithmetic coding of the bits, using the contexts described above, we conclude that the bit-plane coding can achieve performance similar to that of selected rate and distortion pairs from the ECSQ operational rate-distortion curve. The restriction is due to the use of quantization step sizes that vary by a factor of two as each successive bit is encoded. This conclusion is also based on the assumption that the context C_s for the sign bit effectively collapses to depend at most on only the first "1" encountered in the bit-plane scanning, an assumption that appears reasonable in practice.

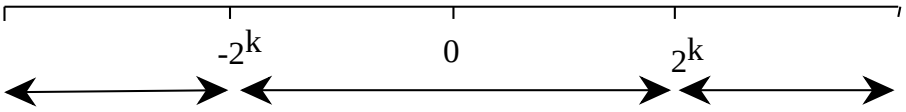

Figure 3: Three-level quantizer implicit with encoding of b_k.

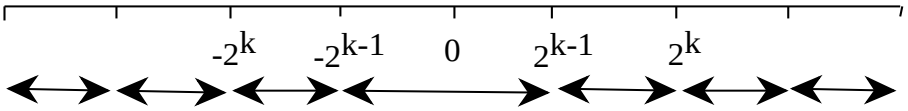

Figure 4: Seven-level quantizer implicit with encoding of b_k, b_{k-1}.

II. C. Modeling of classified coefficients

A detailed analysis of the SPIHT algorithm reveals that it implicitly uses a form of classification. Whenever a coefficient in a tree present in the LIS is found significant at stage n (and moved to the LSP) all neighbors in $B(i,j)$, and any previously insignificant ancestor coefficients in the SOT, are also threshold tested and moved to the LIP or LSP. This is effectively assigning all these coefficients (both significant and insignificant) to a class, say $C(n)$. In every subsequent stage, all coefficients in the LIP are threshold tested (sorting pass), and hence encoded at one bit per coefficient (plus the sign bit encoded if the coefficient is found significant). All coefficients in the LSP are refined, and hence also encoded at one bit per coefficient. Subsequent arithmetic coding can be used to further (although only slightly [2]) reduce the bit rate. The point, however, is that once coefficients are assigned to class $C(n)$, they are all encoded at every subsequent stage.

The classification implicit in the SPIHT algorithm is studied by forming a normalized class database. Nine "natural" greyscale images are used to generate a histogram of the classified wavelet coefficients. The coefficients in each class are normalized by dividing each coefficient by the respective class threshold, resulting in normalized coefficients in the range $(-2.0, 2.0)$. Combining the normalized data from each class yields the resulting histogram shown in Figure 5. The histogram is multi-modal and clearly does not match the generalized Gaussian model previously proposed for classification-based subband image coding [1].

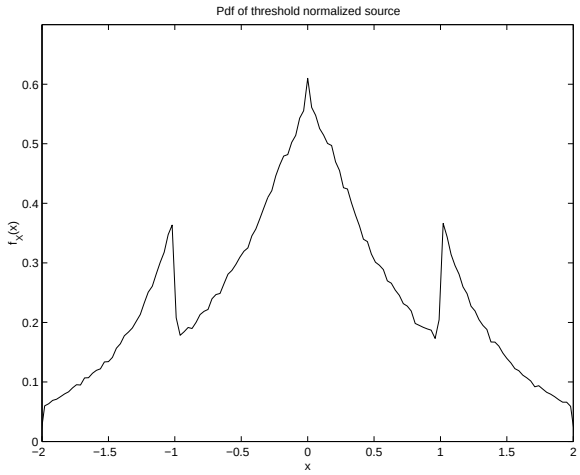

Figure 5: Histogram of threshold normalized coefficients.

II. D. Quantization Performance

The same normalized class data used to create the histogram is also used to evaluate quantization performance. The mean-squared error (mse) distortions reported are expressed as signal-to-noise ratio (SNR), $10\log_{10}\left(\sigma^2/mse\right)$, where σ^2 is the variance of the normalized class data.

The Blahut algorithm [4] is used to compute the first-order rate distortion function for a memoryless source with the marginal probability density function of Figure 5. This provides a convenient reference for comparison with various quantization methods.

Entropy coded scalar quantization (ECSQ) performance is evaluated using uniform threshold quantization (UTQ) [5]. Both uniform and uniform-with-deadzone (at the origin) quantizers are used. For uniform quantization, the reproduction levels are either midpoints or centroids. For the quantization with deadzone, the reproduction levels are either midpoints, centroids, or the estimate of centroids used in the SPIHT algorithm. The results are shown in Figure 3 as distortion (SNR) vs. quantizer output entropy.

Arithmetic and trellis coded quantization (ACTCQ) [6] is used to encode the normalized class coefficients. The encoding performance shown in Figure 6 is for a uniform TCQ codebook with either one or two zero codewords. The latter has been found to be appropriate for low-rate encoding of generalized Gaussian sources with small shape parameter [6]. Centroids are encoded for the two quantizer regions closest to zero, and midpoints are used

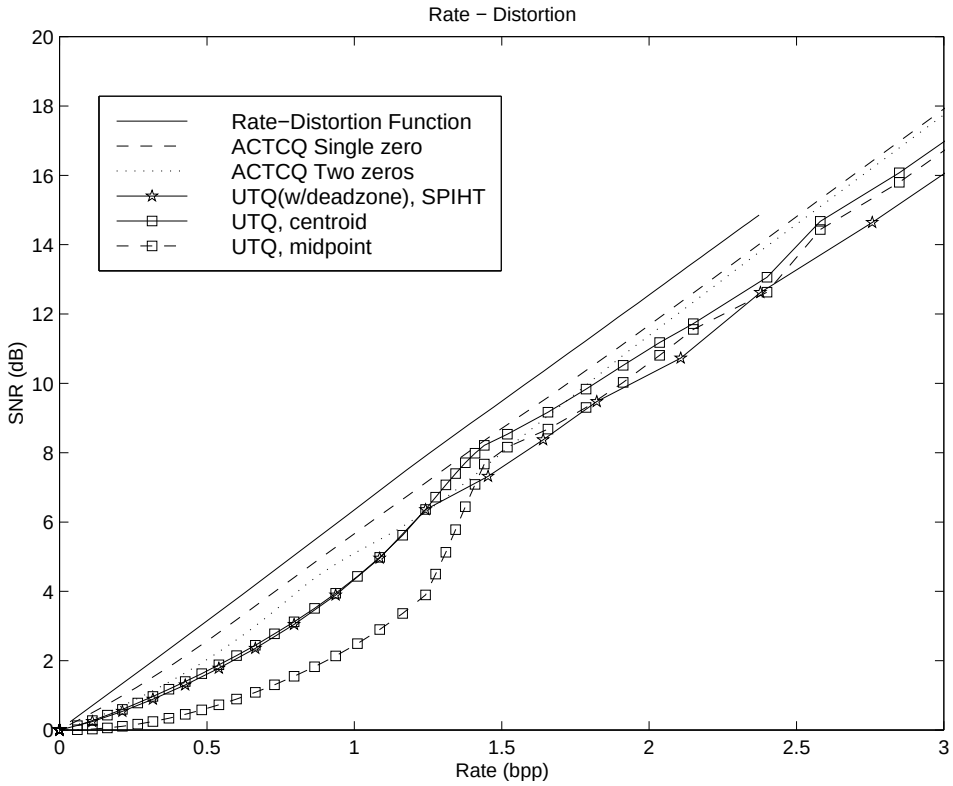

Figure 6: Rate vs. Distortion for several quantizers.

for the remaining regions, as in [8]. The ACTCQ performance is based on encoded file sizes, rather than quantizer output entropy as in the ECSQ results.

Figure 6 indicates that by using the SPIHT algorithm to classify wavelet coefficients, but using a different quantizer than that used in SPIHT, further improvement in coding performance can be obtained. Moreover, close examination of the UTQ results suggests an explanation for the excellent quantization performance achieved by the SPIHT algorithm. Since the number of class coefficients appears to grow roughly exponentially with the classification threshold index [9], at least for relatively small encoding rates, the overall distortion is heavily influenced by the encoding performance for the final class. The SPIHT algorithm classifies in basic 2×2 coefficient blocks, so in addition to the one bit/coefficient comparison required to determine significance, at least one coefficient in four must also have its sign bit coded. If only one coefficient in each block is significant, the average bit rate is 1.25 bits/sample. The UTQ performance curve (using either centroids or the SPIHT reproduction levels) displays a

local maximum relative to the rate-distortion curve at an encoding rate of about 1.25 bits/sample. Hence, for that final class encoded at positive rate, the SPIHT UTQ provides especially good performance. It is also interesting to note that the UTQ without deadzone performance is very close to the ACTCQ performance at a rate of about 1.4 bits/sample when centroids are used. Unfortunately, there appears to be no obvious way to easily modify the SPIHT algorithm to use a uniform quantizer without deadzones due to the successive refinement.

Figure 6 also shows that the TCQ codebook with a single zero codeword performs uniformly better than the two zero codebook. This is quite different from the performance for a generalized Gaussian source [6], and seems to be due to the multi-modal nature of the class density.

II. E. Interband Dependence

The SPIHT algorithm can be viewed as a form of embedded bit-plane coding. The across-subband quadtree structure of the SOTs can exploit interband image dependence in the encoding. Although a precise characterization of the interband dependence remains an open problem, the following simple experiment illustrates how much interband dependence is exploited in the SPIHT algorithm.

Assume the octave decomposition of Figure 1. Index the octave levels, from coarse to fine, as $k = 1, 2, \ldots$. At each level down a SOT there is a corresponding block size, $2^k \times 2^k$. Now, for each subband at level k (except the LFS) perform a (pseudo) random scrambling of the block position within the subband. Subsequent SPIHT encoding imposes SOTs, but without across-band image dependence. That is, by the block position scrambling, any local across-band spatial dependence in a SOT has been destroyed. After SPIHT decoding, by inverting the random block scrambling, a decoded image can be generated.

Table 1 presents SPIHT encoding performance for several images with the block scrambling described above. When the interband dependence is removed for the three natural images, there is a degredation of about 0.1 to 0.4 dB in PSNR performance. For a synthetic image, testimage, there is a much more significant degredation. These results indicate that the SOTs used in SPIHT do indeed take advantage of interband dependence, but that for these natural images only small coding gain is achieved in the SPIHT coding of this dependence.

Image	Rate	PSNR (dB)	
		SPIHT	SPIHT w/ scrambled blocks
Goldhill	0.125	28.48	28.31
	0.25	30.56	30.35
	0.5	33.12	32.80
	1.0	36.55	36.20
Lenna	0.125	31.11	30.68
	0.25	34.14	33.72
	0.5	37.25	36.87
	1.0	40.46	40.15
Barbara	0.125	24.85	24.71
	0.25	27.58	27.41
	0.5	31.39	31.22
	1.0	36.41	36.11
Testimage	0.125	23.91	23.18
	0.25	28.98	28.12
	0.5	37.01	34.83
	1.0	48.38	46.31

Table 1: Impact on coding from block scrambling.

III. Quadtree Classification and TCQ Encoding

Motivated by the preceeding discussion of SPIHT performance, a quadtree classified and trellis coded quantized (QTCQ) wavelet image compression algorithm is described. The quadtree classification is closely related to SPIHT, but uses fewer lists and explicitly forms classes for subsequent TCQ encoding. Arithmetic and trellis coded quantization is described in [6]. Although the results presented here are for an implementation of arithmetic coded TCQ that does not have a progressive decoding property, it could be made so by using the bit-plane coding method of [11]. The encoding algorithm presented uses a quality factor and, like SPIHT, no rate allocation is performed.

III. A. Quadtree Sorting Algorithm

The classification in QTCQ is performed progressively as the image compression occurs. Initially, there is only one class which contains all of the coefficients, $\chi^{(-1)}$. The class $\chi^{(-1)}$ is allocated zero rate. During the first classification pass, class C_0 is created, and its elements are removed from from $\chi^{(-1)}$. There are then two disjoint classes, C_0 and $\chi^{(0)}$, with $C_0 \cup \chi^{(0)} = \chi^{(-1)}$. On the n^{th} pass, C_{n-1} is extracted from $\chi^{(n-2)}$ leaving the $n+1$ classes $C_0, \ldots, C_{n-1}$, and $\chi^{(n-1)}$.

Let $q \geq 0$ be a parameter (a "quality factor"), and let k satisfy $q2^k \leq \max|c_{i,j}| < q2^{k+1}$, where $\max|c_{i,j}|$ is the largest wavelet coefficient magnitude. Define the sequence of thresholds $T(n) = q2^{k-n}, n = 0, 1, \ldots, k$. The $N \stackrel{\triangle}{=} k+1$ thresholds are used to define $N+1$ classes, $C_0, C_1, \ldots, C_{N-1}, \chi^{(N-1)}$. The classification is done by finding, in pass n, all previously unclassified coefficients such that $q2^n \leq |c_{i,j}| < q2^{n+1}$, where n is decremented after each pass. For a given n, the coefficient $c_{i,j}$ is declared *significant* if $|c_{i,j}| \geq q2^n = T(n)$. Note that the final threshold used is $T(k) = q$. The coefficients in the final class $\chi^{(n-1)}$ are all decoded as the zero codeword. The classification algorithm uses a quad-tree[3] structure similar to that in SPIHT [2] to locate significant coefficients. In the n^{th} pass, if any element of a quad-tree, $D(i,j)$, is significant, the four highest elements in the tree, $O(i,j)$, are defined to be in class n. The four elements also become roots for new quad-trees. Each of these new trees is also tested for significance, progressing down the tree until all significant coefficients are found. All coefficients

[3]The terminology "quad-tree" is generic. To accommodate arbitrary sized rectangular images, a parent node can have more, or less, than four offspring.

declared to be in class n are stored in a *list of pixels* (LP).

Figure 1 shows the generic quad-tree structure. Initially, a quad-tree is formed with a root node at each wavelet coefficient in the three bandpass subbands of the coarsest level of the octave decomposition. Groups of four root nodes are linked to an artificial node, becoming the parent of four quad-trees. All of the parent nodes are stored in a list of nodes, called a *list of insignificant sets* (LIS). For the lowest frequency subband (LFS), coefficients are not placed in any quad-trees. A *list of pixels* (LP) is defined for the coding, and initialized with the wavelet coefficients in the LFS.

A function $S_T(\cdot)$ is a significance test on a tree, κ, defined as

$$S_T(\kappa) = \begin{cases} 1, & \max_{(i,j)\in\kappa} |c_{i,j}| \geq T, \\ 0, & otherwise. \end{cases} \tag{2}$$

S_T tests to see if any *descendents* of a node are significant with respect to the threshold, T.

The significance test, lists, sequence of thresholds, and TCQ are assembled into an algorithm for encoding image wavelet coefficients, as follows.

ENCODING ALGORITHM

1. **Initialization:** initialize LP with all $c_{i,j}$ in LFS, LIS with all parent nodes. Output $n = \lfloor \log_2 (\max |c_{i,j}| / q) \rfloor$. Set the threshold $T = q2^n$.
2. **Sorting:**
   - for each node, κ, in LIS do:
     - output $S_T(\kappa)$;
     - if $S_T(\kappa) = 1$ then
       - for each child of κ do:
         - move coefficient to LP;
         - add to LIS as a new node;
       - remove κ from LIS.
3. **Quantization:** for each element in LP, quantize and encode using ACTCQ.
4. **Update:** Remove all elements in LP; replace T by $T/2$; go to step 2.

Figure 7 shows a block diagram for the flow of the classification process. By comparison with Figure 2, the similarities of the QTCQ classification to the SPIHT algorithm are apparent.

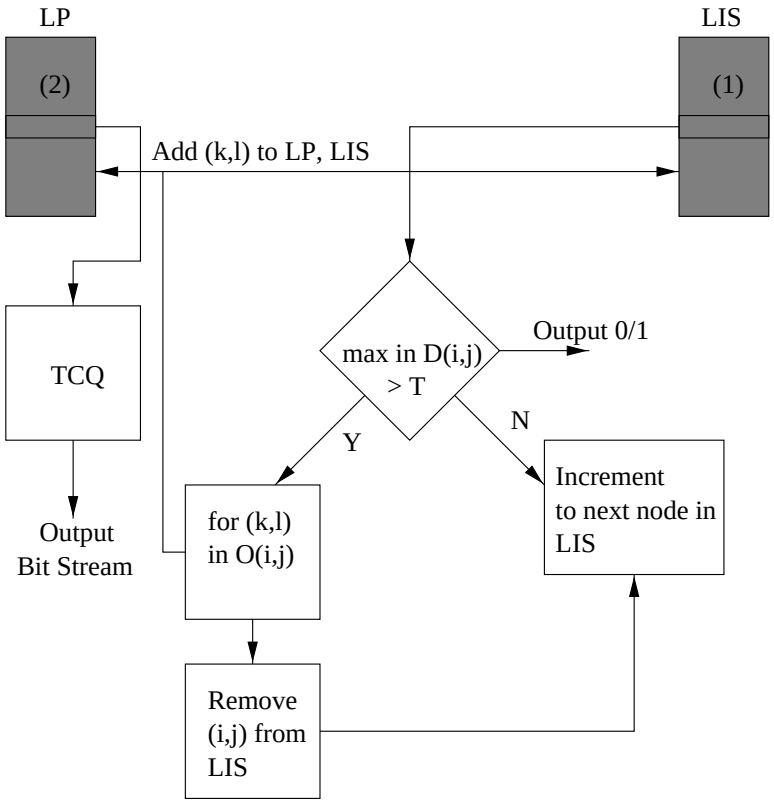

Figure 7: Classification Process in QTCQ.

The decoding algorithm is similar to the encoding algorithm. All outputs above are replaced by inputs, and the ACTCQ encoding is replaced by the ACTCQ decoding process. Since the branching only occurs based upon the S_T data, which are output by the encoder, the decoder can exactly duplicate the encoder's execution path for classification.

In step 3 of the encoding algorithm, an appropiate TCQ codebook step size must be selected. Empirical tests show that using $\Delta = \alpha \cdot q$, $\alpha = 0.7$, for classes $C_0, C_1, \ldots C_{N-1}$ provides most of the TCq granular gain. Using different TCQ step sizes for C_{N-1} and the other classes can provide small additional improvement.

Image	Rate	QTCQ	SPIHT
lenna	0.125	31.3433	31.107
	0.25	34.4278	34.141
	0.5	37.5610	37.247
	1.0	40.8091	40.460
barbara	0.125	25.2902	24.852
	0.25	28.1394	27.579
	0.5	32.0480	31.392
	1.0	37.1547	36.411
goldhill	0.125	28.6842	28.475
	0.25	30.7775	30.557
	0.5	33.4485	33.125
	1.0	36.9938	36.550

Table 2: PSNR performance for several images.

III. B. Performance

The QTCQ encoding algorithm was implemented using a biorthogonal (9,7) wavelet filter bank [15] and adaptive arithmetic code [12] to encode the TCQ indices. Although the encoded bit stream does not have a progressive decoding property, an alternative implementation of the arithmetic coding, using the TCQ bit-plane coding approach in [11] would yield a progressively decodable bit stream.

Table 2 compares the performance of the QTCQ encoding algorithm to SPIHT. The fraction of encoding rate used for the classification map is similar in the two algorithms. The QTCQ improvement in PSNR is due to the granular gain of the trellis code, which for the 8-state trellis code implemented is at most about 1 dB (at high encoding rate) [6].

Figure 8 compares the visual quality of decoded SPIHT and QTCQ images at an encoding rate of 0.5 bits/pixel. The QTCQ decode image typically has (slightly) sharper image details, due to the improved granular fidelity of the trellis code.

QTCQ executable code for various computing platforms is available at www.eecs.wsu.edu/~irl/software.

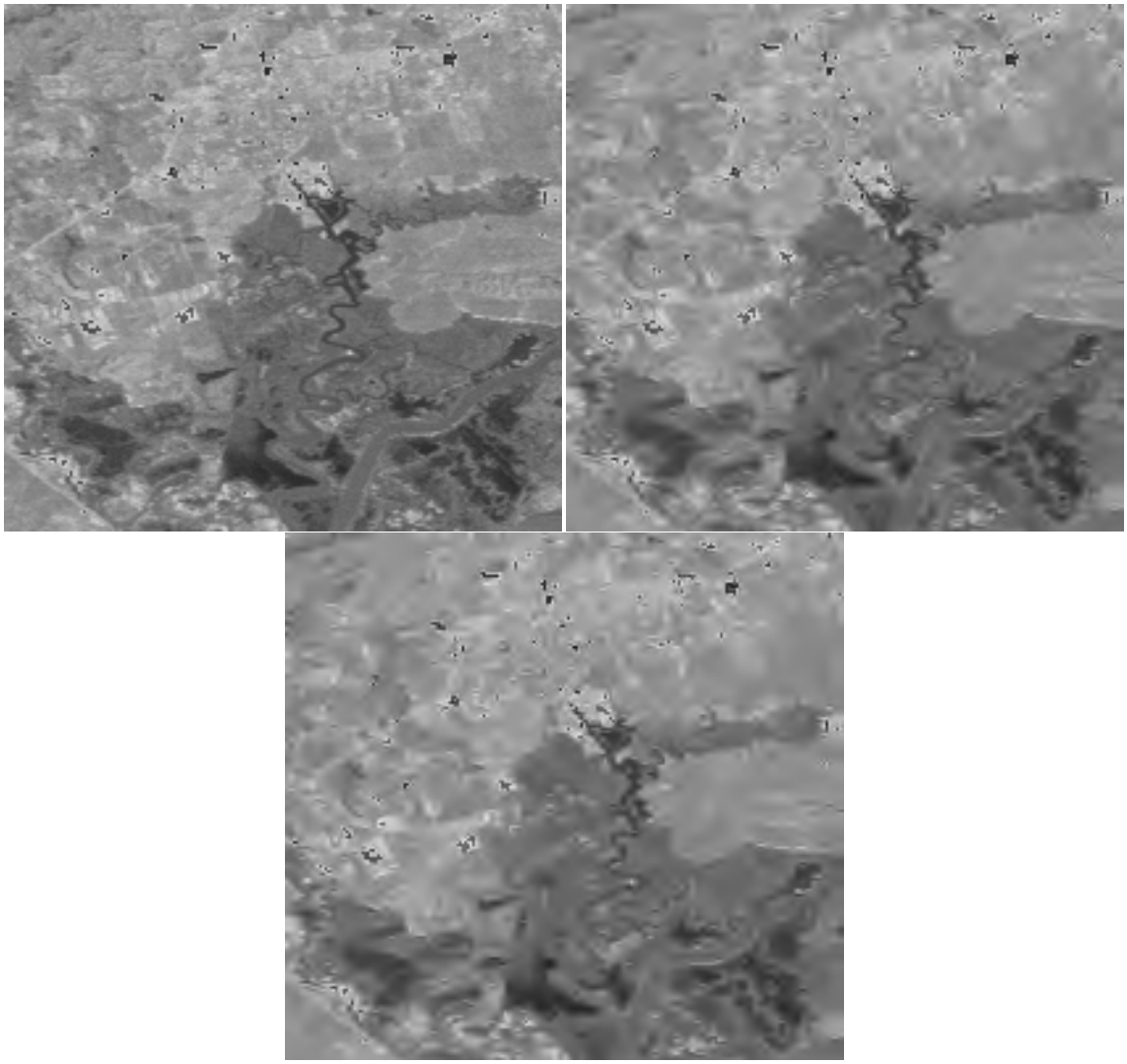

Figure 8: Visual comparison between original 512×512 landsat image, and the image coded by QTCQ, upper right (29.42 dB), and SPIHT at bottom (29.16 dB) at 0.5 bpp.

IV. Extensions and Conclusions

Various extensions of the basic QTCQ algorithm have been considered. An optimized allocation of encoding rate to the separate classes was found to provide negligible improvement to rate vs. distortion performance. Using YCrCb color space, a color image QTCQ compression algorithm was implemented. Small performance improvements were observed (for some images) by structuring the quad-trees across chrominace subbands.

The SPIHT image compression algorithm is a method of bit-plane coding of the (implicitly) uniformly quantized wavelet coefficients. Most of its effectivness is due to the efficient encoding of zeros in the higher frequency subbands. Small additional coding gains are achieved by exploiting inter-subband dependence. However, for many "natural" images, this additional coding gain appears to be small. By using a similar quad-tree algorithm to explicitly classify the wavelet coefficients and trellis coded quantization to encode the classes, further performance improvement is achieved. The improvement, ranging from about 0.5 dB at moderate encoding rates to as much as 1 dB at high encoding rates, is due to the granular gain of the TCQ. At moderate encoding rates, this coding gain is perceptually meaningful as improved granular fidelity in the decoded image.

References

[1] 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," *IEEE Trans. Image Processing*, vol. 6, pp. 1473-1486, Nov. 1997.

[2] A. Said, W. A. Pearlman, "A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," *IEEE Trans. Circuits & Systems for Video Technology*, Vol. 6, pp. 243-250, June 1996.

[3] G. M. Davis and S. Chawla, "Image Coding Using Optimized Significance Tree Quantization," *Proc. Data Compression Conf (DCC)*, Snowbird, UT, March 1997.

[4] R. E. Blahut, "Principles and Practice of Information Theory", Addison-Wesley, 1987.

[5] N. Farvardin and J. W. Modestino, "Optimum quantizer performance for a class of non-Gaussian memoryless sources," *IEEE Trans. Inform. Theory*, vol. 30, pp. 485-497, May 1984.

[6] R. L. Joshi, V. J. Crump, and T. R. Fischer, "Image subband coding using arithmetic and trellis coded quantization," *IEEE Trans. Circ. & Syst. Video Tech.*, vol 5, no. 6, pp. 515-523, Dec. 1995.

[7] M. W. Marcellin, "Wavelet/TCQ image compression," Document NCITS/L3.2/97-062.

[8] J. H. Kasner and M. W. Marcellin, "Universal trellis coded quantization," *IEEE International Symposium on Information Theory*, 1995.

[9] W. Zhang and T. R. Fischer, "TCQ subband image coding to exploit dependence," *Conference Proceedings*, 1996 International Conference on Image Processing, Sept. 1996.

[10] J. M. Shapiro, "Embedded image coding using zerotrees of wavelets coefficients," *IEEE Trans. Signal Processing,* vol. 41, pp. 3445–3462, Dec. 1993.

[11] P. J. Sementilli, A. Bilgin, F. Sheng, M. W. Marcellin, and J. C. Kieffer, "Progressive transmission in trellis coded quantization-based image coders," *Proceedings*, pp. 588-591, 1997 International Conference on Image Processing, Sept. 1997.

[12] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic Coding for Data Compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520-540, June 1987.

[13] W. A. Pearlman, "The Set Partitioning in Hierarchical Trees (SPIHT) Algorithm," *JPEG 2000 Document Regiser*, WG1N692, 1997.

[14] M. W. Marcellin and T. R. Fischer, "Trellis coded quantization of memoryless and Gauss-Markov sources," *IEEE Trans. on Communications*, vol.38, no.1, pp. 82-93, Jan. 1990.

[15] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Proc.*, vol. IP-1, pp. 205–220, April 1992.