Title
Analysis Based Coding of Image Transform and Subband Coefficients

Authors
Algazi, Ralph
Estes, Robert R.

Publication Date
1995

Peer reviewed

Analysis Based Coding of Image Transform and Subband Coefficients

V. Ralph Algazi Robert R. Estes, Jr.

CIPIC, Center for Image Processing and Integrated Computing
University of California, Davis

ABSTRACT

Image coding requires an effective representation of images to provide dimensionality reduction, a quantization strategy to maintain image quality, and finally the error free encoding of quantized coefficients. In the coding of quantized coefficients, Huffman coding and arithmetic coding have been used most commonly and are suggested as alternatives in the JPEG standard. In some recent work, zerotree coding has been proposed as an alternate method, that considers the dependence of of quantized coefficients from subband to subband, and thus appears as a generalization of the context-based approach often used with arithmetic coding.

In this paper, we propose to review these approaches and discuss them as special cases of an analysis based approach to the coding of coefficients. The requirements on causality and computational complexity implied by arithmetic and zero-tree coding will be studied and other schemes proposed for the choice of the predictive coefficient contexts that are suggested by image analysis.

Keywords: analysis based modeling, wavelet coding, zerotree coding, image modeling, context–based modeling, embedded zerotree wavelet (EZW) code.

1 INTRODUCTION

Lossy encoding of images often consists of three stages: representation, quantization and error-free encoding. Block-based transforms (such as the DCT of JPEG) and subband and wavelet decompositions are commonly used to convert an image into a representation with good energy compaction. The transform coefficients are then quantized to reduce the information and achieve the desired bitrate. The quantized coefficient image is then entropy encoded in a lossless fashion. We have shown (for wavelets) that good results can be obtained with such a framework.[1,2] Recently, however, more sophisticated techniques have surfaced which, in some sense, analyze the image to exploit higher level correlations that exist in the transform domain. One such technique, which has received a lot of attention, is the embedded zerotree wavelet (EZW) code.[5]

In this paper, after introducing lossy image compression, we discuss the EZW code, analyze its behavior, and then propose a model which uses simple, but more conventional, modeling techniques to achieve better performance. As such, we show that the zerotree data structure, from a coding efficiency perspective, is of little use.

2 LOSSY IMAGE COMPRESSION

2.1 Representation

Images display a high degree of correlation, i.e., they are low pass in nature. Usually, the first stage in an image compression system involves some sort of transformation to decorrelate the data and concentrate the energy into a few coefficients. Common methods include block based transforms, such as the DCT used in JPEG, and joint spatial–frequency based decompositions as used in subband and wavelet coding.

From an appropriate viewpoint, block based DCT codes can be interpreted as a subband coding techniques — all one has to do is reorder the data so that coefficients which share the same frequency band are grouped together. A major problem, however, is that in block based codes there is no interaction between pixels in different blocks which, when coupled with coarse quantization, results in blocking artifacts. Subband and wavelet techniques decompose the image into frequency bands and, because they are filtering based approaches, do not suffer from blocking artifacts and typically generate higher quality images at low bitrates. In wavelet transforms, we use a hierarchical decomposition which recursively decomposes the LL band into LL, HL, LH and HH subbands (since the image energy is concentrated there). These subbands are critically subsampled so that the number of samples remains the same after the transformation. In this work, we will restrict our attention to wavelet based encoders.

2.2 Quantization

Typically, the number of samples resulting from image transformations remains the same, but the precision required to specify the transform coefficients increases. Often, the output of the representation is a set of real-valued coefficients, which we cannot encode with a finite number of bits. Thus, quantization is required to reduce the coefficients to finite precision. Furthermore, quantization is often the only way we can reduce the information content of the source in a controlled fashion.

In all common transformations, there is some notion of frequency in the transform domain, and better quantizers exploit the human visual system by quantizing higher frequencies, where errors are less visible, more coarsely than lower frequencies. Theoretically, vector quantization (VQ) results in better performance, but is much more complex to implement. Furthermore, the gain of VQ over scalar quantization (SQ) is reduced for decorrelated (transform) coefficients. Recent wavelet transform encoding techniques attempt to exploit the benefits of VQ, while minimizing the computational burden.[3]

2.3 Modeling

We typically divide the coding process into two components: modeling and error–free encoding. The goal of modeling is to predict the distribution to be used to encode each pixel. Error–free encoding will be discussed in the next section.

In coding, our goal is to encode each sequence of symbols $\{s_1, s_2, \ldots, s_n\}$ with $-\log_2 p(s_1, s_2, \ldots s_n)$ bits. If the symbols are independent and identically distributed (i.i.d), this reduces to $-n \log_2 p(s)$, where $p(s) = p(s_i), \forall i$. If not, then modeling is the task of estimating $p(s_1, s_2, \ldots s_n)$. Clearly, even for a binary source, the number of possible sequences is unmanageable and simplifications must be made. Reformulating the problem as that of estimating a set of conditional probabilities, i.e.,

$$p(s_1, s_2, \ldots s_n) = \prod_i p(s_i | s_1, \ldots, s_{i-1}) \tag{1}$$

does not reduce the complexity, but does lead to a related formulation which is useful in practice. To reduce the

inherent complexity, we re-express (1) as

$$p(s_1, s_2, \ldots s_n) = \prod_i p(s_i | f(s_1, \ldots, s_{i-1})), \tag{2}$$

where f is some unknown but to be determined function. If f is the identity mapping, then both formulations are identical. Our goal however, is to reduce the problem of estimating the symbol distributions to a manageable size. This is accomplished by restricting the range of f to a small set of *states*. Now, we can associate with each state a conditional source comprised of all symbols which occur in that state. The success of our model, is determined by the extent that the conditional sources are decorrelated with one another and are i.i.d. random processes. Unfortunately, the determination of the best f for a given source is extremely difficult and we must resort to heuristic techniques and intuition.

What is our intuition? First, we have at our disposal all previously transmitted symbols. Typically, due to complexity constraints, we must also reduce the range of f. This is done by choosing a set of pixels (a context) from the set of all previously transmitted pixels. This choice is of utmost importance, since our prediction (classification) is based entirely upon these pixels. For raster scan techniques, this typically corresponds to neighboring pixels to the left and above the current pixel. These should be the pixels that supply the most information about the current pixel. Often, f is taken to be the identity function on this reduced set of pixels, so that's its design consists entirely of their selection.

Note that in the above discussion, we have mentioned previously transmitted pixels. This suggests that the order in which we scan the data is also of importance, and adds another level of complexity to image encoding. Causality is an ill-defined property in 2–D, especially when a frame buffer is available. Hierarchical pyramids, for instance, correspond to a reordering of the data which, hopefully, has desirable properties. Finally, we realize that, in some fashion, the probabilities determined by the encoder must be communicated to the decoder so that the data can be correctly decompressed.

2.4 Error-free codes

Given a set of probabilities for each conditional source, encoding is straight-forward. One possibility is to design a Huffman code for each conditional source. The problem with Huffman codes, however, is that they cannot encode highly skewed sources efficiently. Therefore, one has to resort to source extension techniques, such as run-length coding, to obtain good results. Furthermore, highly skewed sources are the rule, not the exception, in coding applications. As an alternative, one can use arithmetic codes which have many advantages. They can encode a source at a rate arbitrarily close to the entropy of the source, a single encoder can encode multiple interleaved conditional subsources, and they can be easily made adaptive. Their only fault is that they are more computationally complex than Huffman codes (which can be implemented as table lookups). Much work has been done to make arithmetic codes more efficient, by approximating the required multiplications and developing efficient probability estimation schemes.

Often it is more effective, or at least requires less resources in the decoder, to use simpler models (which do not predict the symbols as well) and allow the probabilities to adapt to the source statistics. This technique is effective for slowly changing subsources. If the statistics vary rapidly, then developing a better predictor (f) is often a better idea. The arithmetic codes commonly used in practice are adaptive. One may pay a small penalty using an adaptive code, but the class of sources which can be encoded efficiently is much larger than when static probability estimates are used.

When using both Huffman and arithmetic codes, the (conditional) probability tables must be communicated to to the decoder. If the characteristics of the source are well defined, then the statistics can be precomputed from an ensemble of typical images and stored at (built into) both the encoder and decoder. Such systems will not be able to efficiently encode data which lies outside the design parameters. As an alternative, the probabilities can be transmitted to the decoder, implicitly, as is done in adaptive Huffman and arithmetic coders. The advantage of this latter technique is that universal codes (which, asymptotically, can encode any source optimally) can be designed.

3 THE EMBEDDED ZEROTREE ENCODER

The EZW code encodes images, in an "embedded" fashion, from their dyadic wavelet representations. The goal of embedded coding is to generate a single encoded bitstream which can be truncated, to achieve any desired rate, and used to reconstruct the best possible rendition at that rate. In a sense, it circumvents the quantization stage discussed earlier. Upon closer examination, however, we will see that this is not the case; quantization now consists of determining the best order in which to transmit the source symbols. Shapiro[5] attacks this problem in an amplitude first fashion — wavelet transform coefficients with the same magnitude are assumed to have equal importance, and should be transmitted before coefficients with smaller magnitudes. It is beyond the scope of this paper to describe all the intricacies of the EZW coder. We present a high level description which, hopefully, is sufficient to understand the sequel. More details can be found elsewhere.[5]

The EZW coder encodes wavelet transform images with respect to a set of decreasing thresholds, $T_0 > T_1 > \cdots > T_{N-1} = T_{min}$. Only pixels which exceed the current significance threshold are processed during each pass. Typically, $T_i = T_{i-1}/2$ and the EZW coder is very similar to bitplane encoding techniques. We will restrict our discussion to this simplified case, for which we present two alternate descriptions: one due to Shapiro in terms of lists and a second based on the idea of significance maps (binary images that specify which pixels are currently significant ($> T_i$)).

As described by Shapiro,[5] the EZW coder maintains two lists, a dominant list and a subordinate list. Initially, all pixels are placed on the dominant list in a predefined order. In the dominant pass, this list is scanned and the location of all pixels whose magnitude exceeds the first significance threshold, T_0, and their signs, are encoded using the zerotree data structure. These pixels are then transferred to the subordinate list, and the corresponding coefficients in the wavelet transform image set to zero so that their location is not encoded again in later passes. In the subordinate pass, the next bit in the representation of each pixel on the subordinate list is encoded. The subordinate list is then sorted (using only the information that is known at the decoder), and the process repeated for each threshold until a bitrate target is met or T_{min} is reached.

When the thresholds can be expressed as $T_i = M2^k$, for integer k, as we have assumed, the algorithm can be restated more succinctly in terms of the bitplanes of the integer image obtained by uniformly quantizing the wavelet transform image with step size T_{min}. First, we decompose the quantized image into its sign and magnitude components. Next, we encode the most significant magnitude bitplane, and the corresponding sign bits, using the zerotree data structure. The significance map (a binary image which indicates which pixels are significant with respect to the current threshold) is then updated as the union of its previous value and current magnitude bitplane. It serves as an indicator function, indicating which pixels are significant and must be refined. We then proceed to the next bitplane and encode the bits which are specified by the significance map. These bits are set to 0, and we iterate, as before, using the current bitplane as the most significant one.*

The zerotree data structure is of central importance in the EZW code. Its importance lies in its ability to efficiently encode large blocks of zeros in the significance maps, and its exploitation of the hierarchical correlation in wavelet transform images. At its simplest, a zerotree represents a binary (dyadic wavelet significance map) image as a set of quadtrees, rooted at each pixel in the lowest resolution HL, LH and HH subbands, which contain all pixels with the same frequency orientation and spatial location. To complete the tree, each pixel in the LL band is defined to be the parent of the pixels at the same location in the lowest HL, LH and HH bands. Shapiro elected to encode the sign bits as part of the zerotree and chose a 4 symbol representation to encode it: zerotree root (`0`), isolated zero (`z`), positive significant (`+`) and negative significant (`-`). A zerotree root is used to indicate that the entire subtree rooted at the corresponding node is zero (or insignificant), and allows an efficient description of large all white blocks. The isolated zero symbol is used to indicate that a pixel is not significant, but that one of its children is. Significant symbols are classified as positive or negative significant. An example significance map (augmented with isolated zeros, zerotree roots and sign information) and its corresponding zerotree are shown in Figure 1.

A zerotree is encoded by encoding the symbols encountered on a predetermined path through the corresponding

*This interpretation is more intuitively pleasing than that using lists, but is not quite accurate. To implement sorting of the subordinate list, a list has to be maintained.

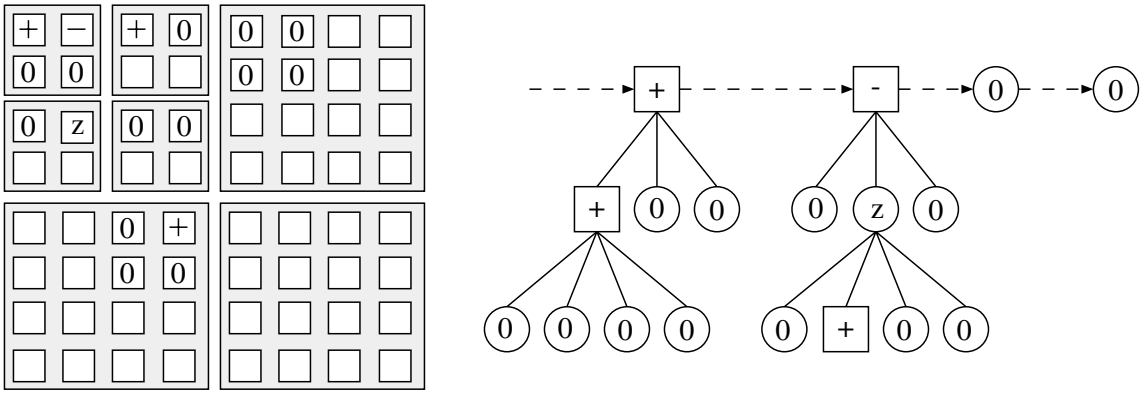

Figure 1: Zerotree representation of a 2 level dyadic wavelet decomposition with 4 significant pixels.

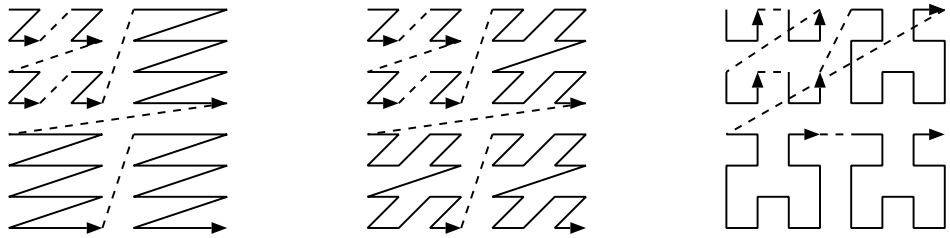

Figure 2: Example scanning paths for a 2 level dyadic wavelet decomposition. Left to right: raster scan, Morton order, and Peano scan. The dotted lines define the connectivity between subbands.

augmented significance map. Some example orderings are shown in Figure 2. The only restriction placed on them is that a pixel parent (in the zerotree) should be scanned before the pixel. This corresponds, roughly, to a depth first traversal of the zerotree. Alternatively, one could just encode an in order traversal of the tree, but there may be correlations that are better exploited by more specific scanning patterns. The chosen ordering is important consideration with respect to the embedded nature of the algorithm, and emphasizes the fact that, for images, there really is no well-defined notion of causality. Intuitively, is would seem that we should transmit the lower frequency components first.

For entropy encoding, Shapiro conditions the zerotree symbols using the significance of a pixels parent and the previous pixel in the defined ordering. An isolated zero cannot occur at the leaf nodes, so that a ternary alphabet can be used for the highest frequency bands. The bits encoded in the subordinate pass are encoded in a single context, without any conditioning. All subsources are encoded using an adaptive arithmetic code[8] with a maximum frequency count of 256.

To completely specify the EZW coder, as described, we must also specify the set of wavelets, the normalization used in the wavelet transform, a scanning order, and the minimum significance threshold, T_{min}, or M.

3.1 Implementation

We have implemented the EZW coder, as described above, except that we use the binary arithmetic QM-coder as our entropy encoder. Thus, we do not have control over the adaptivity of the encoder (since it is built in) and must map the multi-alphabet sources onto binary trees before encoding. Furthermore, we have made many parts of the algorithm optional so that we can evaluate their contribution to its performance. For all results presented, we use a 6-level dyadic wavelet decomposition of Lena, based on the biorthogonal 9/7 wavelets of Barlaud, which we,[1,2] and others,[7] have found to be useful in coding applications.

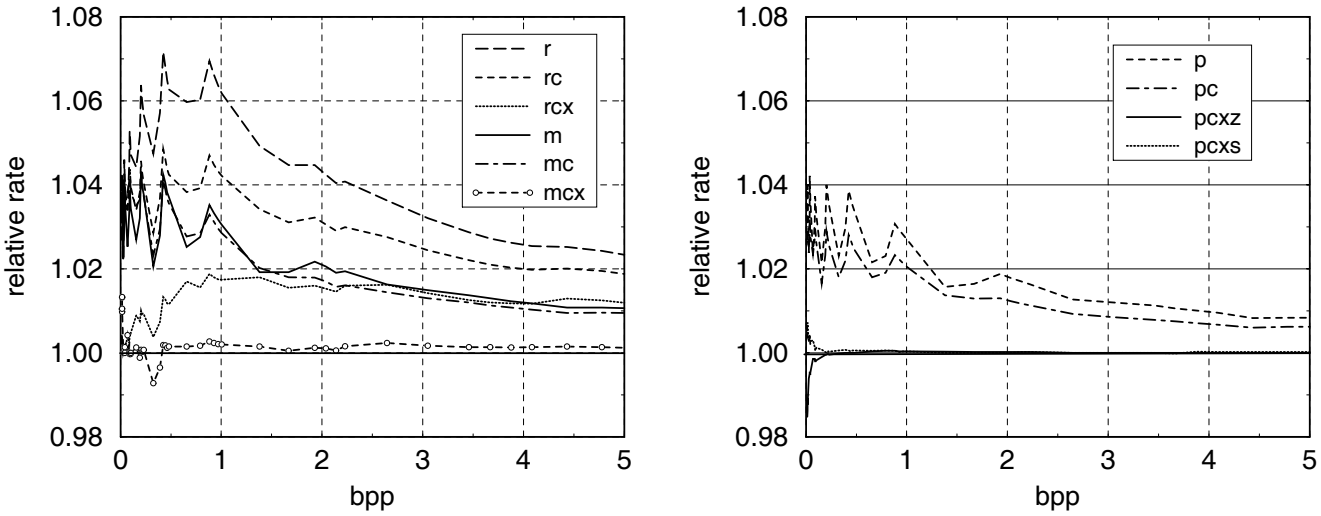

Figure 3: Usefulness of various options in the design of a zerotree based encoder. All comparisons are in terms of the output bitrate relative to `pcx`.

3.2 Performance of the EZW coder

Many of the options in the EZW coder appear to be of questionable value. In this section, we present results which characterize the algorithm and emphasize its important features.

In our first experiment, we analyzed the effects of scanning pattern (`[rmp]`,[†] for raster, Morton, and Peano scans), 2 pixel conditioning of the zerotree symbols (`c`, for conditioning), subtracting and separately transmitting the mean of the LL band (`z`, for zero mean), sorting the pixels on the subordinate list (`s`, for sorting), and the cost of encoding a significance map of all significant pixels instead of just the new ones (`x`, for the removal of significant pixels from the significance map, so that they are not encoded in subsequent passes). The results are presented in Figure 3, where all results are relative bitrate measurements, with respect to the `pcx` results. First, we note that mean removal and sorting have negligible effect. Mean removal results in a small gain and sorting a small loss at the lowest bitrates. Conditioning is important, but less so for the more spatially localized scanning patterns (Morton and Peano scans). Encoding just the new pixels in significance maps (option `x`) buys around 2% overall. Summarizing, the scanning order, conditioning of the zerotree symbols and the zeroing of pixels transferred to the dominant list are important (a few percent in terms of bitrate), while mean removal and sorting have little value.

An example of the performance (PSNR) of the zerotree algorithm as a function of bitrate for a constant T_{min}, as well as the contribution of the various components,[‡] is given in Figure 4. We see that most of the bits are used to encode the significant pixel location information and simple calculations show that we get very little compression of the sign and (subordinate pass) refinement bits. Note the scalloped nature of the PSNR curve (as compared to the linear progression of the binary symbols encoded versus *bps* plot), especially in later passes. This is an artifact of the embedded quantization strategy. The peaks in the performance occur at the end of passes, when the same quantization has been applied to the entire image. This would suggest that, in the later passes, there is a significant cost incurred to describe the first few levels of the zerotree, without matched benefit.

In the our last experiment analyzing the characteristics of the EZW code, we consider the effect of T_{min} and

[†] We use the [·] regular expression notation to indicate one a choice of options.

[‡] These curves are obtained by dumping the probabilities estimated by the QM-encoder and computing conditional point-wise entropies. The actual bitrates are about 7% larger.

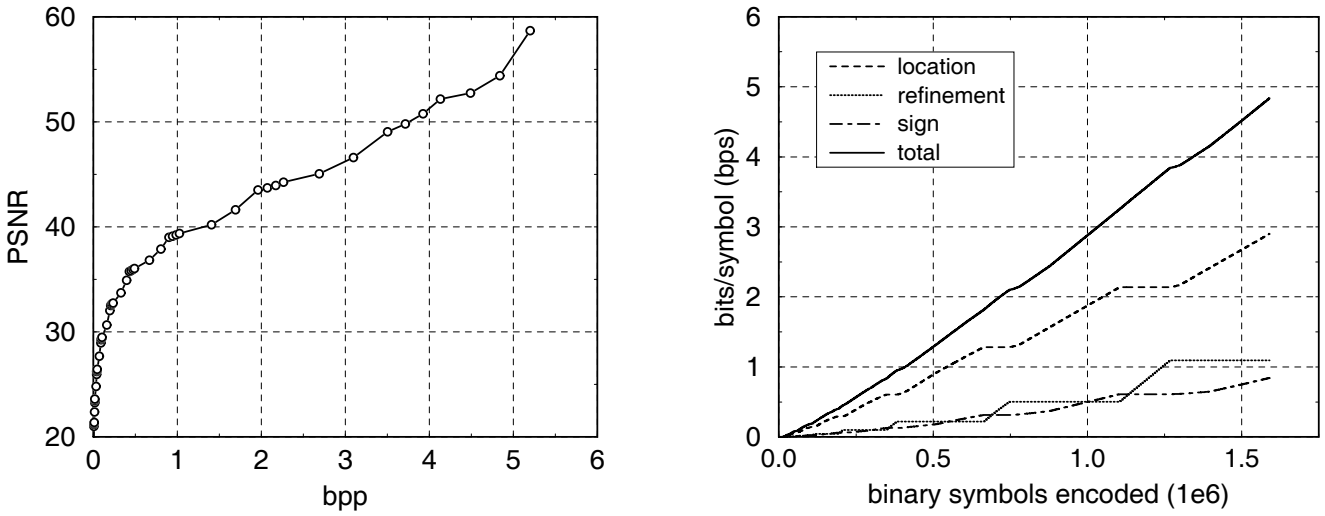

Figure 4: Left: performance of EZW `rcx`. Right: corresponding bitrate by category.

the wavelet normalization strategy. Varying T_{min} causes the peaks to shift, and the resulting performance at a given bitrate can vary by more than 1 dB (see Figure 5). This emphasizes the fact that, even though the EZW coder is an embedded algorithm, to obtain (its) optimum performance at a given bitrate, a costly optimization (over T_{min}) is required. This optimization is similar to common quantizer design approaches encountered in more standard encoding frameworks.

Although never explicitly mentioned in Shapiro's paper, the normalization factor chosen in the forward and inverse wavelet transforms is very important. Given a a normalization factor s, we scale the results by $1/s$ and s, respectively, in the forward and inverse 1-D wavelet transforms. With this definition, a normalized transform, in which the range of the transform is commensurate with the range of the original image, corresponds to $s = \sqrt{2}$. With $s = 1$, we get an amplification of the LL band coefficients by a factor of 4 for each iteration in the wavelet decomposition. Recall that the EZW code uses an amplitude first decomposition, therefore, this range expansion corresponds to a reordering of the data to be encoded and has a significant effect on its performance. It can also be interpreted as a frequency weighted quantization method (see DISCUSSION). Our experimental results are shown in Figure 5, where we see that values between 0.8 and 1.2 are fairly interchangeable, but that outside this range performance degrades quickly.

4 ANALYSIS BASED WAVELET CODING

Shannon's coding theory suggests that we can encode i.i.d. random variables at rate arbitrarily close to their entropy. If the source is correlated, then we should be able to compress it more efficiently. Examining the output of wavelet transform images, it is clear that structure has survived the transform. It has been shown[3] that there is little correlation amongst the (high band) coefficients, but that correlation exists between the magnitude of these coefficients. The EZW code exploits this correlation, but maybe not as efficiently as possible.

In this section, we start by analyzing the EZW code to clarify the correlations which it exploits. Next, we present a simple alternative code, which does not use the zerotree data structure, exploits similar information, and performs better than the EZW code, suggesting that there is no inherent advantage to using the zerotree data structure for encoding wavelet coefficients. A properly chosen model performs better. Finally, we present several simple extensions to our model which show that further gains are achievable. We leave for future work

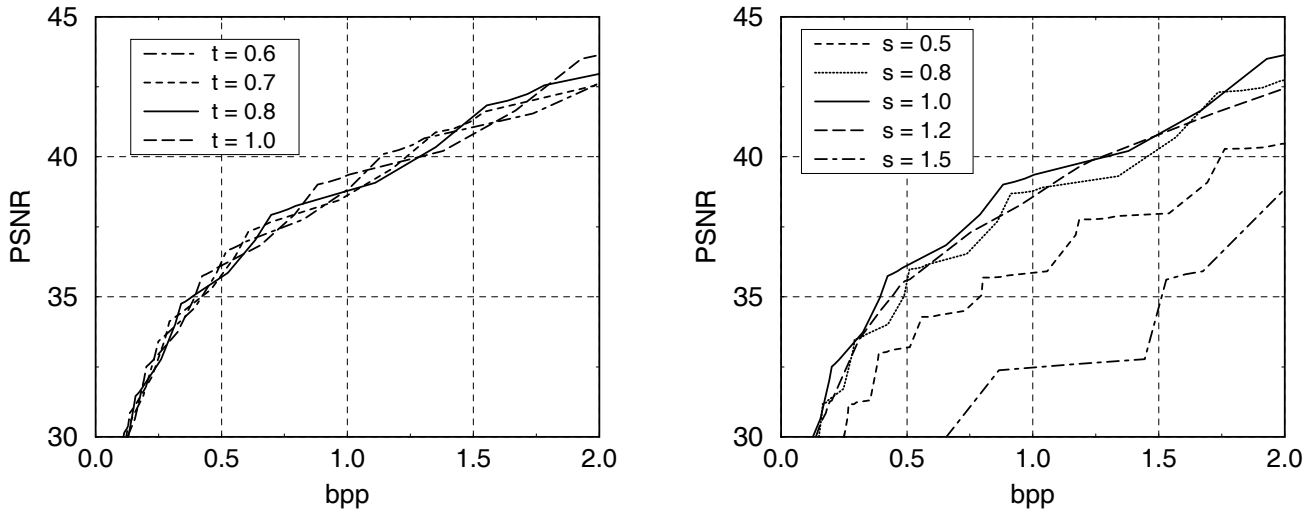

Figure 5: Left: performance as a function of T_{min}. Right: as a function of wavelet normalization factor. Both results are for EZW `pcx`.

the determination of optimum encoding contexts.

4.1 A simple EZW–like code

For our first model, we wish exploit the same features as the EZW code, yet not rely on the zerotree data structure. Instead, we depend upon sound modeling principles, i.e., upon the intelligent choice of conditioning contexts.

As for explicit dependencies, the EZW uses the significance (or insignificance) of its parent and the previous pixel (in the chosen scanning order) to condition zerotree symbols. Furthermore, the removal of pixels after they have been added to the subordinate list is equivalent to predicting the current significance map by the previous significance map. Thus, we also condition pixels by their values in the previous significance map. From elementary coding theory, conditioning is a better choice than prediction. In this case, conditioning leads to two sources: $p(\cdot|0)$ and $p(\cdot|1)$, while the predictive code reduces this to a single source with $q_0 = p_0 p(0|0) + p_1 p(1|1)$ and $q_1 = p_0 p(1|0) + p_1 p(0|1)$, where q_i are the probabilities used by the predictive model, p_i the probabilities from the previous significance map, and $p(\cdot|\cdot)$ the conditional probabilities of pixels in the current significance map, given their value in the previous significance map. Note, that $p(1|1) = 1 - p(0|1) = 1$, due to the nature of the significance maps. The gain due to this conditioning may be small , especially near the beginning of the sequence when $p(0|0)$ may be fairly close to 1.

In the EZW code, a separate context is used for the highest frequency bands since only three symbols are required instead of four. Alternatively, we use the scale of the wavelet subband as part of the conditioning context. Recall that the sign bits are encoded as part of the zerotree, so that they are encoded in the same conditioning context as the location bits. We augment this to include the sign of the significant pixels, not just their significance, in an attempt to reduce the overhead of transmitting the sign information. Similarly, for the refinement bits, EZW uses a single context, but we choose to use contexts similar to our dominant pass contexts, in an attempt to exploit additional correlation.

Using regular expressions to summarize the set of contexts, for dominant and refinement decisions we use `[dr][1-6][is][is]` and for sign decisions we use `s[1-6][0+-][0+-]`, where `d` and `r` represent the dominant

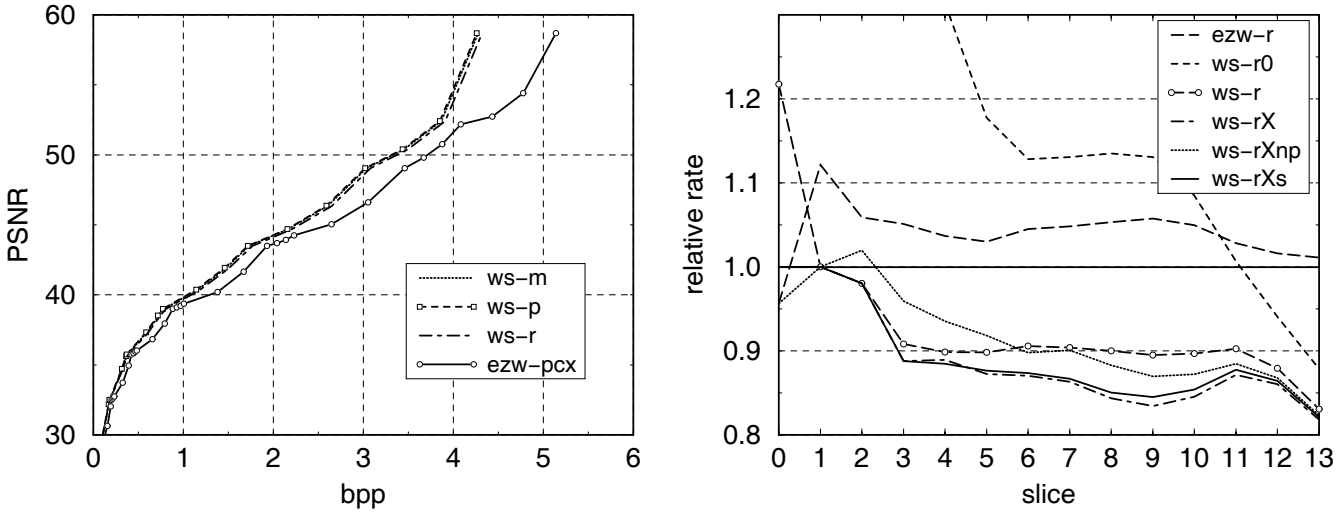

Figure 6: Analysis bases models for wavelet transform image encoding. Left: performance of simple models versus `ezw-pcx`. Right: Relative performance of extended models versus `ezw-rcx`.

and refinement contexts, respectively, `1-6` is the scale (or frequency band) in the transform, `i` and `s` are the significance and insignificance of the parent and previous pixels, and `0`, `+`, and `-` are used to represent their signs. Thus, we use a total of 102 binary subsources. Note that we no longer have multi-alphabet sources, as when encoding a zerotree; all symbols are binary.

The code simply consists of scanning each plane, in the prescribed order, and encoding the symbols with respect to the corresponding contexts described above. We no longer have dominant and subordinate passes as in the EZW code. We present the results, with $T_{min} = 1$, a wavelet normalization factor of 1 and our three scanning patterns, in Figure 6. From the curves we see that this simple model, indeed, outperforms the EZW code, while preserving the embedded nature of the code, further emphasizing that the gain is primarily to due modeling, not the zerotree data structure. To be fair, more than twice as many binary decisions must be encoded by the new scheme.[§] Using a Peano scan can reduce the bitrate by as much as 2%. Although we have not explicitly presented it here, the gains we have obtained are almost entirely in the encoding of the location information, suggesting that our extended contexts to sign and refinement bits are of little use.

4.2 Extensions

An extensive analysis of correlations in the transformed image would be required to determine the optimum contexts for encoding each decision in the embedded code. However, as a step in the right direction, we present some simple extensions to the above model to indicate that further gains are possible, if more pertinent modeling is employed.

We present 4 modifications to the model used above (with a raster scan ordering), which we denote `ws-r`. In `ws-rX` (for extended), we add two additional conditioning pixels in the same band, representing the significance of the neighboring pixels in the previous significance map to the right and below the pixel currently being encoded. In `ws-rXs`, we augment `ws-r` by adding 2 sibling pixels, that is, pixels at the same location, but in alternate frequency bands. For example, if we are encoding a pixel in the LH_2 band,[¶] we use the corresponding pixels in

[§] But we have made no attempt to minimize this computation burden.

[¶] We number our subbands incrementally from the lowest to highest frequencies.

the HL_2 and HH_2 bands. If the pixels have not been scanned in the current bitplane, we use their significance from the previous bitplane. Given that our scanning order is $\ldots, HH_i, HL_{i+1}, LH_{i+1}, HH_{i+1}, \ldots$, for our example, we would use the current plane for HL_2 and the previous plane for HH_2. Model `ws-rXnp` is the same model as `ws-rX`, except that the parent pixel is removed from the context, to assess its predictive value. Due to the increased number of states in our extended contexts, we divide the subbands into low and high frequency bands, instead of conditioning by scale, to keep the number of states at a reasonable level. And lastly, model `ws-r0` is meant to put these results into perspective. It uses no conditioning whatsoever, i.e., a single context is used to encode all binary decisions.

The results of these experiments are shown in Figure 6. Before making comparisons, we need to explain the horizontal axis. It represents the last pass (bitplane or slice) that has been processed by each algorithm. Since the EZW code uses less symbols than does our model, we must normalize the results. It is important to note, however, that the end of each dominant pass corresponds exactly with the end of each pass in our model and, at these points, both coders are encoding *exactly* the same quantized wavelet transform coefficients. Thus, we are justified in making our comparisons solely based on bitrate. The corresponding bitrates for `ezw-rcx` are 0.0007, 0.0010, 0.0016, 0.0030, 0.0066, 0.0153, 0.0386, 0.0914, 0.204, 0.428, 0.898, 1.96, 3.50, and 5.20 *bpp*, respectively.

From the figure, we see that the extended sources do, indeed, perform better than our EZW–like model (annotated with open circles in the plot), and that the two neighbors in the current band offer a better context than the siblings do, but not much better. Furthermore, the parent pixel accounts for up to 3% of the performance over `ezw-rcx`. We also note the the very simple 0-order model performs within 13% of `ezw-rcx` for bitrates greater than .04 *bpp*, and performs better than it at the highest bitrates. Thus, conditioning results in a bitrate reduction of about 20% over the interesting bitrate range (say .05-2 *bpp*). Over this same range our codes perform 10-15% better than the corresponding zerotree results, `ezw-rcx`. Comparing the conditional entropies after slice 13 is encoded with the actual encoded bitrates, we found that arithmetic encoding improves the EZW based results about 4-6% and our conditional wavelet source results by 5-10%, but reduces the 0-order context bitrate by over 40%. This gain is primarily due to the large areas (typically at high frequencies) where the source consists almost entirely of zeros, to which the adaptive arithmetic code adapts to and encodes efficiently.

5 DISCUSSION

We have shown that there are no inherent advantages to using the zerotree data structure, with the exception of there being less decisions to encode with the binary arithmetic encoder. Using simple model design techniques a more efficient model can be developed that does not rely on complicated data structures. This is somewhat intuitively pleasing, since in some respects the zerotree is like a block based code, which has been shown to be inferior to conditional codes of the same complexity.[4]

Furthermore, we have shown that additional gains are possible by defining better conditioning contexts. The zerotree does not exploit the dependencies between pixels in neighboring blocks effectively, and it does not exploit the dependencies between siblings in the representation at all. The extent of the gains possible are not known, and the results we obtained were modest. Intuitively, however, we see significant structure in the wavelet transform images, and techniques which use higher level descriptions of the edges at lower resolutions to predict higher frequency bands appear to have merit. We leave this for future work.

Although it claims to eliminate the need for quantization, their is an underlying quantization mechanism used by the EZW code, as we have implied earlier. The choice of wavelet normalization and the set of slice thresholds defines a set of quantizers. Indeed, the EZW code specifies a (heuristically designed) set of quantizers, and a method for interpolating between them. Although never explicitly mentioned in the EZW discussions, a HVS weighted quantizer could as easily be applied to the wavelet transform before the bitplanes are encoded. This would reorder the data transmitted by the code, and could be made more effective from a HVS viewpoint.

6 ACKNOWLEDGMENTS

This research was supported in part by the UC MICRO program, Pacific Bell, Lockheed and Hewlett Packard.

7 REFERENCES

[1] Jian Lu, V. Ralph Algazi, and Robert R. Estes, Jr. Comparison of wavelet image coders using the Picture Quality Scale (pqs). In *Proceedings of the SPIE, Wavelet Applications II*, volume 2491, pages 1119–1130, April 1995.

[2] Jian Lu, V. Ralph Algazi, and Robert R. Estes, Jr. A comparitive study of wavelet image coders. Submitted to *Optical Engineering* special issue on VCIP, 1995.

[3] Michael T. Orchard and Kannan Ramchandran. An investigation of wavelet-based image coding using an entropy-contrained quantization framework. In Storer and Cohn,[6] pages 341–50.

[4] J. J. Rissanen and G. G. Langdon, Jr. Universal modeling and coding. *IEEE Transactions on Information Theory*, 27(1):12–23, January 1981.

[5] Jerome M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *Signal Processing*, 41(12):3445–3462, December 1993.

[6] James A. Storer and Martin Cohn, editors. *DCC'94:Data Compression Conference*, Snowbird, Utah, March 1994. IEEE Computer Society Press.

[7] John D. Villasenor, Benjamin Belzer, and Judy Liao. Filter evaluation and selection in wavelet image compression. In Storer and Cohn,[6] pages 351–60.

[8] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Comm. Assoc. for Computing Machinery*, 30(6):520–39, June 1987.