Tree-structured vector quantization with significance map for wavelet image coding

Pamela C. Cosman, Sharon M. Perlmutter†, Keren O. Perlmutter†

4-178 EE/CSci, 200 Union St. SE, University of Minnesota,
Minneapolis, MN 55455, e-mail: cosman@ee.umn.edu
†Information Systems Lab, Stanford University, Stanford, CA
94305-4055, e-mail: sharonp@isl.stanford.edu, keren@isl.stanford.edu

Abstract

Variable-rate tree-structured VQ is applied to the coefficients obtained from an orthogonal wavelet decomposition. After encoding a vector, we examine the spatially corresponding vectors in the higher subbands to see whether or not they are "significant," that is, above some threshold. One bit of side information is sent to the decoder to inform it of the result. When the higher bands are encoded, those vectors which were earlier marked as insignificant are not coded. An improved version of the algorithm makes the decision not to code vectors from the higher bands based on a distortion/rate tradeoff rather than a strict thresholding criterion. Results of this method on the test image "Lena" yielded a PSNR of 30.15 dB at 0.174 bits per pixel.

1 Introduction

The discrete wavelet transform combined with scalar or vector quantization has led to numerous algorithms for image compression. Using a multiresolution framework, the wavelet transform organizes the coefficients to enable effective quantization and encoding. The temporal/scale tessellation of the wavelet transform avoids the blocking artifacts that are common for schemes using only temporal tessellation (e.g., JPEG and VQ used without transforms).

Wavelet encoding involves taking the discrete-time wavelet transform of an image and quantizing the wavelet coefficients based on some bit allocation scheme. *Bit allocation* is the process of assigning a given number of bits to a set of different sources (e.g. wavelet subbands) to minimize the overall distortion of a coder. For scalar quantization, the bit allocation scheme chooses the size of the quantizer step, or bin, for each subband. Among the best wavelet coding reported to date has been the embedded zerotree wavelet (EZW) algorithm [9], which uses scalar quantization together with an efficient scheme for indicating where the coefficients of large magnitude are located. For vector quantization, the bit allocation scheme controls the vector dimension and the number of codewords for each subband. After quantization the indices are usually entropy encoded. Image compression using vector quantization (VQ) applied to wavelet coefficients has been the focus of many recent studies. Lattice VQ, full search VQ, entropy-constrained VQ, and finite-state VQ have been tried, with good results.

In this paper, we quantize wavelet coefficients using variable-rate tree-structured vector quantizers designed by the generalized Lloyd algorithm, combined with a simplified version of the zerotree significance map idea for indicating where vectors of large coefficients are located. A brief review of tree-structured vector quantization is given in Section 2. Section 3 describes the way in which this technique is combined with a zerotree structure in this work. Results and conclusions are presented in Section 4.

2 Tree-structured vector quantization

Tree-structured vector quantization (TSVQ) is an image compression technique that is rapid for both the encoder and the decoder. A binary TSVQ consists of a tree with nodes labeled by candidate reproduction vectors. An input vector is compared to the labels of the two child nodes available at the root node, and the node with the minimum distortion label (the nearest neighbor) is selected. The encoder then performs a similar test for the new node's children and continues in this manner until a terminal node is reached. The label of the terminal node is the final reproduction, and the binary vector describing the sequence of encoder decisions is the codeword stored or sent to the decoder. The decoder then performs a table lookup to produce a local reproduction. A TSVQ is thus described by a tree (nodes and labels) and a distortion measure used to select nodes in a nearest neighbor fashion (see, e.g., [3]).

A balanced TSVQ is grown one level at a time using, for example, the splitting method of the generalized Lloyd algorithm [4]; this results in a fixed rate code. A variable rate code can be implemented by an unbalanced tree, obtained either by growing a balanced tree and then pruning it back so that it becomes unbalanced, or by "greedily" growing an unbalanced tree directly [6]. The latter algorithm is an extension to VQ of a common decision tree design technique [1].

Given a tree T, assume that we split one of its leaves (terminal nodes) j into two new leaves j_L and j_R. The centroids of the two new leaves are determined by the generalized Lloyd algorithm. Let $\Delta D = D' - D$ and $\Delta R = R' - R$ be the change in the distortion and rate, respectively, due to splitting j. The distortion of a node is the average squared error between the centroid of the node and the training vectors mapping into that node. Then the ratio of the change in distortion to the change in rate due to splitting leaf j is

$$\lambda = -\frac{\Delta D}{\Delta R}, \tag{1}$$

which is the "goodness of split" for leaf j.

As in decision tree design, we can design a TSVQ one node at a time, always splitting the node with the largest λ. The algorithm is "greedy" in the sense that each node is split without considering its later effect on the tree. This method results in an unbalanced tree, because the node that is split can be at any depth. There will be more codewords available to code high distortion events; this is where the tree will have been split the most. Therefore, unbalanced trees are able to code high distortion events at a higher resolution than can balanced trees which are limited by their initial depth.

The growing method optimally trades off rate and distortion for each new node in a greedy fashion. The resulting tree can then be pruned with the generalized Breiman, Friedman, Olshen, and Stone (BFOS) algorithm [1]. One can achieve a lower distortion for a given average rate by optimally pruning the tree with the generalized BFOS algorithm rather than by removing the nodes in the reverse order in which they were added. This is because the growing algorithm is greedy, whereas the BFOS pruning algorithm removes nodes optimally.

3 TSVQ with zerotree significance maps

The zerotree wavelet algorithm of Shapiro [9] uses scalar quantization. A wavelet coefficient x is said to be insignificant with respect to a given threshold T if $\mid x \mid < T$. The zerotree algorithm is based on the hypothesis that if a wavelet coefficient at a coarse scale is insignificant with respect to a given threshold T, then all wavelet coefficients of the same orientation at the same spatial location at finer scales are likely to be insignificant with respect to T also. Given a threshold level T, a coefficient x is said to be an element of a *zerotree* for threshold T if itself and all of its descendants are insignificant with respect to T. Figure 1 shows the arrangement of a coefficient and its descendants. An element of a zerotree is a *zerotree root* if its parent is not an element of a zerotree. When wavelet coefficients are to be coded at low bit rates by scalar quantization followed by entropy coding, the zero symbol will be the most probable symbol after quantization, and a binary *significance map* can be used to indicate the location of the non-zero values. The EZW algorithm uses the zerotree structure as an efficient way to compress significance maps. EZW is an iterative algorithm that refines the quantization step by one-half at each iteration and results in an embedded code. The quantization step used at each iteration applies to all of the subbands, so only the original quantization step needs to be stored as overhead. The iterations repeat until all of the available bits are allocated. Adaptive arithmetic coding is applied to the bit stream that results from this process.

There have been a couple of previous efforts to make use of the zerotree structure as part of a vectorial quantization scheme for wavelet coefficients. In one approach, a coefficient from a coarse band is grouped with 4 coefficients from the next finer band of the same orientation, and with the 16 corresponding coefficients of the finest-scale band, to produce a 21-dimensional vector [5]. Those vectors which have all components below a specified threshold are designated zerotrees and not coded. The remaining coefficients are re-organized into lower dimensional vectors, and vector

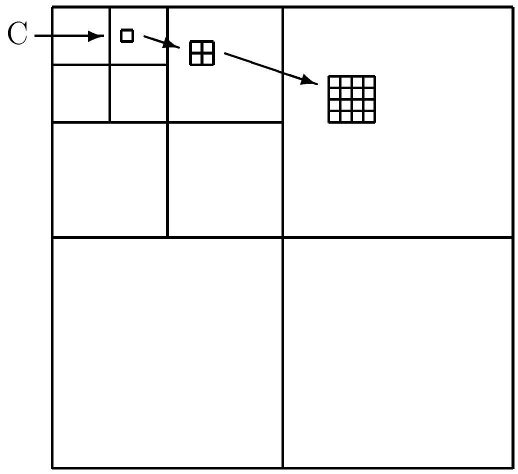

Figure 1: A coefficient "C" and its descendants

quantized. An algorithm that is closer to the iterative refinement approach of Shapiro uses a multistage lattice to progressively refine the vectors of coefficients [7]. Each input vector is coded with a series of vectors of decreasing magnitudes. At each stage, the orientation of the reconstruction vector is selected from a finite set of unit energy codevectors, which are chosen based on a regular lattice.

The algorithm described here differs from these. In our work, the Daubechies orthogonal 8-tap filter was used to decompose the image to 4 levels [2], producing a total of 13 subbands. Different vector sizes and shapes were used in the different levels. The lowest band was coded by scalar quantization (1×1 vectors). The finest scales were encoded using 4×4 vectors, and intermediates scales employed vectors of size 2, 4, or 8. Figure 2 shows the decomposition together with the size and shape of vector used in each subband.

A training sequence was composed of 10 images from the USC database. Each image was transformed, and the various bands blocked into vectors of the size and shape chosen for that band. The image "Lena" was used as a test image, and was not part of the training sequence. A large tree-structured vector quantizer was greedily grown on each training sequence, and optimally pruned back.

Bit allocation: Fewer bits can be allocated to the high frequency subbands than to the low frequency ones, as they possess smaller variance, corresponding to less information. Senoo and Girod examined the optimal allocation of bits among the various subbands [8], and their method is employed in the current work. They showed that the minimum overall distortion for a given total rate is achieved when the individual distortion rate curves $D_i(R_i)$ are of equal slope, that is

$$\frac{\partial D_i(R_i)}{\partial R_i} = -\lambda, \tag{2}$$

for all subbands i. This condition is referred to as "Pareto optimality" in economics.

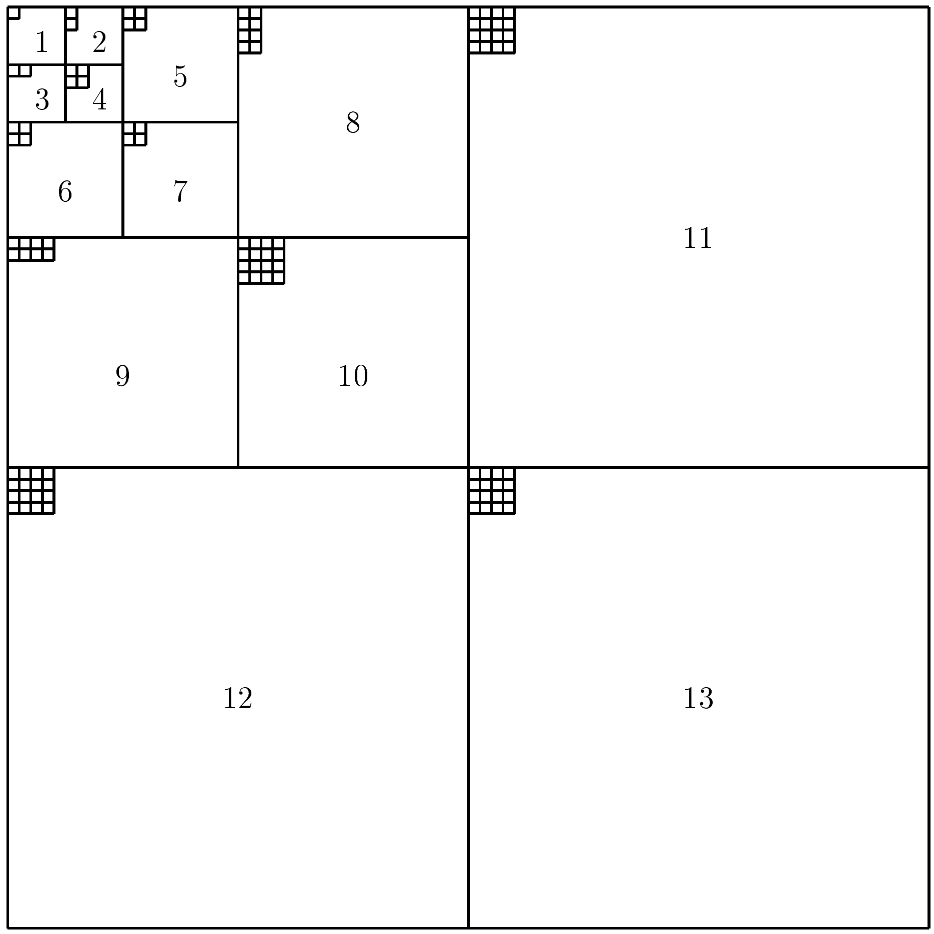

Figure 2: The 4-level wavelet decomposition produces 13 subbands. Here are shown the size and shape of the vectors chosen for each subband.

Each TSVQ was grown out to a large size and then optimally pruned back. For each band, the distortions and rates for the sequence of pruned subtrees provided the $D_i(R_i)$ curves on which we found the points of equal slope. This is only an approximation to the optimal bit allocation because the distortions and rates of the pruned subtrees are computed on the training sequence, not on the test sequence. It is also approximate because only a discrete set of optimally pruned subtrees is available, and these subtrees may be spaced far apart in rate, which may mean the available slopes are widely separated. So by using only these optimally pruned subtrees, one is not actually obtaining equal (or nearly equal) slopes in the different bands.

Zerotrees: A simplified version of the zerotree significance map was incorporated into this algorithm as follows. The lowest subband is encoded without examining future bands. When subband 2 is encoded, a given 2-D vector x is in spatial correspondence with two 4-D vectors of subband 5. Each of these 4-D vectors is in spatial correspondence with two 8-D vectors of subband 8. And each of these 8-D vectors is in spatial correspondence with two 16-D vectors of subband 11. Thus 2 vectors from subband 5, 4 vectors from subband 8, and 8 vectors from subband 11 are all considered to be the descendants of the vector x of subband 2. The vector x is encoded to the reproduction vector $\hat{x}$. If the magnitudes of all the components of vector $\hat{x}$ are below some threshold T_1 then we will transmit one bit to the decoder to inform it of whether this entire collection of descendants is significant with respect to some threshold T_2. If either component of $\hat{x}$ has magnitude greater than T_1, then we do not transmit any side information about the significance of its descendants. Because the decoder receives the encoded vectors of subband 2, it knows the value of $\hat{x}$, and therefore knows whether the extra bit is being sent or not. The threshold T_1 is used for deciding when to spend the extra bit. The threshold T_2 is used for determining whether the descendants can be declared insignificant or not. In this study, T_1 values of 96, 64, and 32, and T_2 values of 48, 32, and 16 were examined. When subband 5 is encoded, those vectors which were previously marked as insignificant will not be coded. The other vectors will be coded. Such vectors arise either because no extra marking bit was transmitted for the parent at all, or because the marking bit for the parent indicated that the descendants could not be ignored. If a vector in subband 5 is encoded, then components of its reproduction are similarly compared to T_1 to see whether or not its descendants in subbands 8 and 11 should be examined for insignificance. For the finest subbands (11, 12, 13) the vectors have no descendants, and thus no extra bits are sent. At low bit rates, the optimal bit allocation scheme may determine that no bits should be allocated to subbands 12 and 13, in which case the zerotree bits for subbands 9 and 10 need not be sent. The extra bits corresponding to the significance map can be entropy coded (as can the bits corresponding to the tree encoding), although in this study entropy coding has not been implemented.

Dispensing with the thresholds: Using thresholds introduces the danger that the thresholds may not be well suited to a particular test image, and that the results may be radically different for different choices of the thresholds. An improved version of the algorithm dispenses with the thresholds entirely. The first threshold T_1 can be eliminated simply by choosing to examine the descendants of *all* encoded vectors. This is equivalent to setting T_1 to be infinite. The second threshold T_2 can be eliminated by choosing the zerotrees according to a distortion-rate tradeoff, rather than by a strict thresholding criterion. If one considers, for instance, a 2-D vector in subband 2, its two 4-D descendant vectors in subband 5, and all their descendants in subband 8 and 11, one has a collection of 170 coefficients. There are a number of ways to code these 170 coefficients. For example, the lone ancestor vector could be encoded and the decoder could be told that its descendants constitute a zerotree, without consideration of what threshold level would actually make those descendants a zerotree. This would lead to a certain distortion D_1 and rate R_1 associated with the entire collection of 170 coefficients. Or the two children vectors in subband 5 could be encoded, and *their* descendants could be declared a zerotree, which would lead to a distortion D_2 and a rate R_2 for the whole set of 170 coefficients. Or the descendants of only one of the children vectors in subband 5 could be declared a zerotree, but not the descendants of the other. This would lead to additional possibilities for D and R. In all, there are 26 ways that the group can be encoded, and each choice produces a (D, R) pair. These pairs can plotted in the D, R plane and the convex hull of the set can be extracted. This curve, which we call the zerotree choice curve, then represents the various optimal distortion/rate tradeoffs obtainable for the group of 170 coefficients by different choices of where to root the zerotrees. We recall that bits were allocated among the subbands by finding points along the $D(R)$ curves for each subband that had equal slope. This same slope can be used to guide the selection of the operating point along the zerotree choice curve. In this way, groups of coefficients are zeroed out not because they meet some strict thresholding criterion, but because zeroing them out makes sense according to a distortion/rate tradeoff. And in fact, it is the same distortion/rate tradeoff that is used to choose the vector quantizer subtree for encoding each subband.

4 Results and Conclusions

The test image "Lena" was decomposed 4 levels using the Daubechies orthogonal 8-tap filter. The subbands were blocked into vectors, and each vector was encoded by the pruned subtree for its subband. With the thresholds T_1 and T_2 set at 64 and 32 respectively, the image encoded to 0.187 bits per pixel (bpp) with a PSNR of 30.0 dB. Those thresholds provided about the best PSNR at that bit rate. The method that did not use thresholds outperformed the threshold method at all bit rates examined. In particular, PSNRs of 29.2, 30.15, and 30.6 were obtained at bit rates of 0.13, 0.17, and 0.21 bpp, respectively. The original image is shown in Figure 3, and an encoded image is shown in Figure 4. These results do not include any entropy coding on the resulting bit stream.

When the test image was encoded using TSVQ without examining the significance of later coefficients (equivalent to just taking $T_1 = 0$) a PSNR of 30.0 dB was obtained at 0.27 bpp, and a PSNR of 28.8 was obtained at 0.183 bpp. This means that using the zerotrees saved about 1/3 of the bits when the PSNR was held to be the same, and improved the PSNR by about 1.2 dB, when the bit rate was held to be the same. The vector quantization that used the zerotree consistently performed better than the one that did not.

We note that the 13 training sequences used *all* of the training vectors from the corresponding subbands. The training sequences therefore were not quite representative of the test vectors that were later encoded, since when the zerotrees are used, fewer test vectors of low magnitude are encoded. That is, the training sequences were appropriate for the regular TSVQ case, and the TSVQs with zerotrees performed well despite this handicap.

VQ applied to wavelet coefficients has been shown by several researchers to produce excellent quality images at low bit rates. The contribution of the current work is two-fold: first, it demonstrates that good quality at low bit rates can be obtained using variable-rate tree-structured VQ on wavelet coefficients. Secondly, and more importantly, the use of the simplified vectorial version of the effective zerotree strategy for eliminating groups of insignificant coefficients provided more than a 1 dB

Figure 3: Original "Lena" image

Figure 4: "Lena" encoded at 0.174 bpp with a PSNR of 30.15 dB

improvement over the same wavelet-TSVQ that did not use it. One can speculate therefore that some of the recent excellent results of wavelet-VQ applied to intra-band vectors could be improved upon by the incorporation of this simple technique, although the technique would be of less use for wavelet-VQ methods that employ cross-band vectors.

In this study, the ranges of various parameters in this algorithm have not yet been explored. Only one filter has been tried (Daubechies 8-tap), at only one depth of decomposition (4 levels), with only one set of choices for the different vector sizes and shapes in the various bands (as shown in Figure 2). One aspect of continuing this work is to explore the results obtainable by varying these different parameters, and by using entropy coding. The basic structure of the vector quantizer could be changed as well.

Acknowledgment: This work was supported in part by a postdoctoral fellowship from the National Science Foundation.

References

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

[2] I. Daubechies. *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[3] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, 1992.

[4] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Trans. Comm.*, COM-28:84–95, Jan. 1980.

[5] I. Moccagatta and M. Kunt. VQ and cross-band prediction for color image coding. In *Proceedings of the Picture Coding Symposium (PCS '94)*, Sept. 1994.

[6] E. A. Riskin and R. M. Gray. A greedy tree growing algorithm for the design of variable rate vector quantizers. *IEEE Trans. Signal Process.*, 39:2500–2507, Nov. 1991.

[7] D.G. Sampson, E.A.B. da Silva, and M. Ghanbari. Wavelet transform image coding using lattice vector quantisation. *Electronics Letters*, 30(18):1477–1478, Sep. 1994.

[8] T. Senoo and B. Girod. Vector quantization for entropy coding of image subbands. *IEEE Transactions on Image Processing*, 1(4):526–532, October 1992.

[9] J. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, December 1993.