Image Coding Using Optimized Significance Tree Quantization

Geoffrey M. Davis and Sumit Chawla
Math, Computer Science Departments
Dartmouth College, Hanover, NH 03755
{*gdavis,chawla*}*@cs.dartmouth.edu*

Abstract

A number of recent embedded transform coders, including Shapiro's EZW scheme, Said and Pearlman's SPIHT scheme, and Xiong *et al.*'s EZDCT scheme employ a common algorithm called significance tree quantization (STQ). Each of these coders have been selected from a large family of significance tree quantizers based on empirical work and *a priori* knowledge about transform coefficient behavior. We describe an algorithm for selecting a particular form of STQ that is optimized for a given class of images. We apply our optimization procedure to the task of quantizing 8x8 DCT blocks. Our algorithm yields a fully embedded, low-complexity coder with performances from 0.6 to 1.9 dB better than baseline JPEG for standard test images.

1 Introduction

A number of recent image coding schemes, including Shapiro's embedded zerotree wavelet scheme (EZW) [4], Said and Pearlman's set partitioning in hierarchical trees scheme (SPIHT) [3], and Xiong *et al.*'s EZDCT scheme[6], all employ an embedded significance mapping and quantization technique that we refer to as significance tree quantization (STQ).

The EZW, SPIHT, and EZDCT schemes are members of a large, general family of embedded, tree-structured significance mapping schemes. We analyze the performance of such schemes below, interpreting them both as crude entropy coding schemes and as constrained vector quantization schemes. We describe a family of random processes that are related to simple models of subband coefficients in images for which these embedded schemes are optimal. Our analysis provides a partial explanation for the excellent performance of these cited coding schemes.

The EZW, SPIHT, and EZDCT schemes make use of significance mapping schemes chosen based on some *a priori* assumptions about transform coefficient behavior in images. We describe a procedure for finding an optimized significance map given a training set of images. We use this procedure to obtain quantization schemes based more on actual image data and less on assumptions about the form of image data. We employ our optimization procedure for 8x8 DCT blocks and obtain a fully embedded, low-complexity scheme that yields substantial improvements in PSNR over baseline JPEG. Without using any entropy coding, our scheme also outperforms optimized JPEG [5] and a version of EZDCT that uses no entropy coding.

2 Significance Tree Quantization

2.1 Significance Trees

Significance tree quantization is a simple scheme for quantizing an n-vector $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ of coefficients using a tree-structured significance map. The basic data structure in an STQ scheme is a tree or set of trees in which each leaf corresponds to one of the scalars X_k. We classify nodes according to their height in the tree, writing $\mathbf{N}_k^{(j)}$ for the k^{th} node in the

tree at height j from the leaves of the tree. We denote the leaves of the tree by $\mathbf{N}_m^{(0)}$. The height of a node $\mathbf{N}_n^{(j)}$ with children $\{\mathbf{N}_{n_1}^{(j_1)}, \ldots, \mathbf{N}_{n_k}^{(j_k)}\}$ is given by $1 + \max(j_1, \ldots, j_k)$. The significance trees in EZW are quadtrees, whereas the trees in SPIHT and EZDCT possess an alternating 4-fold and 5-fold splitting structure. We will consider more general trees that will be determined by an optimization procedure.

We call a coefficient X_k *significant* with respect to a given threshold T if $|X_k| \geq T$; otherwise we say it is *insignificant*. A coefficient is *newly significant* if $2T > |X_k| \geq T$, i.e. if it is significant with respect to T but not with respect to $2T$. A node $\mathbf{N}_k^{(j)}$ is significant if it or any of its descendents are significant; otherwise it is insignificant. Similarly, $\mathbf{N}_k^{(j)}$ is newly significant if it is significant with respect to T but not with respect to $2T$.

Zerotree quantization proceeds iteratively, producing at each stage a map of all coefficients that are newly significant with respect to thresholds T_0, $\frac{T_0}{2}, \frac{T_0}{4}, \ldots$. The initial threshold T_0 is chosen so that $2T_0 > \max\{X_i\}$. The significance map at each stage is generated by the recursive function `NodeSignificance()` which takes as its arguments a current node and a current threshold. Iteration k of the zerotree coding algorithm consists of a call to `NodeSignificance()` with current node equal to the root $\mathbf{N}_0^{(R)}$ and the current threshold equal to $T_0 2^{-(k-1)}$.

`NodeSignificance()` produces a significance map of the current node and all its children with respect to the given threshold. It first tests the significance of the current node. If the node is insignificant, the procedure emits a '0' and then returns. There is nothing more to code in this case, since the current node's insignificance means that all its children are insignificant as well. If the node is newly significant, the procedure emits a '1' and then calls `NodeSignificance()` for all the children of the current node. If the node is significant, but not newly significant, the procedure emits nothing, but calls `NodeSignificance()` for all the children of the current node. In this case there is no need to emit the significance of the current node since it is known from prior iterations of the algorithm. Pseudocode for the algorithm is as follows:

`NodeSignificance (current node` $\mathbf{N}$`, current threshold` T `)`

- `If` $\mathbf{N}$ `is insignificant with respect to` T`, emit '0' and return;`
- `If` $\mathbf{N}$ `is significant with respect to` T `, emit '1';`
- `Call NodeSignificance() for each child of` $\mathbf{N}$ `with threshold` T`;`
- `Return;`

The result of iterating this procedure K times is a sequence of 0's and 1's that bracket the magnitudes of all coefficients into one of the intervals $\{(T_0 2^{-k}, T_0 2^{-k+1}]\}_{1 \leq k \leq K}$ or $[0, T_0 2^{-K}]$. We can compress this output using an adaptive arithmetic coder of some kind[1], or for low complexity applications the output can be used as is. We now describe an embedded procedure for quantizing coefficients once their magnitudes have been specified.

2.2 Embedded Scalar Quantization

The above procedure generates an embedded significance map for the coefficients. We couple this mapping with an embedded quantization scheme, performing one pass of the quantization for every pass of the significance map. This quantization procedure is used in the EZW, SPIHT, and EZDCT schemes.

After the first pass of `NodeSignificance()` all coefficients are known to be either newly significant or insignificant, and all coefficients are known to lie in the interval $(-2T_0, 2T_0)$. We first emit a bit indicating the sign of each newly significant coefficient. With the significance map and these signs the decoder can determine which of the intervals $(-2T_0, -T_0]$ (newly significant and negative), $(-T_0, T_0)$ (insignificant), or $[T_0, 2T_0)$ (newly significant and positive), contain each coefficient. If we were to decode the quantized coefficients at this stage, we would decode them to the centers of their respective intervals, namely to $-\frac{3}{2}T_0$, 0, and $\frac{3}{2}T_0$.

After the second pass of the significance mapping we emit signs of the coefficients that are newly significant in the second pass. This refines the interval $(-T_0, T_0)$ occupied by the coefficients insignificant after pass 1 into into the 3 subintervals $(-T_0, -\frac{T_0}{2}]$, $(-\frac{T_0}{2}, \frac{T_0}{2})$, $[\frac{T_0}{2}, T_0)$. We refine the already significant coefficients by emitting a bit to indicate whether they lie on the left or right half of their respective intervals. If we were to dequantize coefficients at this stage, we would decode them to the centers of their respective intervals. Subsequent passes of the quantization proceed in a similar fashion. The result is an embedded uniform quantizer with a double-width deadzone.

2.3 Significance Test Amortization

We now address the question of why this type of significance mapping is effective. We motivate the use of significance tree quantization from two perspectives. It provides amortization of significance test costs, and it performs a constrained form of vector quantization

At the simplest level, significance tree quantization provides a means of efficiently coding random variables having fractional bit entropies without having to resort to arithmetic coding. Consider the problem of entropy coding a collection of independent, identically distributed random variables X_i that take on values 1 or 0 with probabilities p and $1-p$, respectively. Suppose that 1's are rare, i.e. $p \ll 1$. The simplest approach is to use a bit to specify the value of each X_i, but of course this is inefficient since the entropy of each random variable, $H(p)$, will be considerably less than 1.

When p is sufficiently small we can reduce the coding cost by aggregating the X_i's into sets. We test the entire set for significance and only test the individual X_i's if the set is significant. Here a set of X_i's is significant if any X_i in the set is equal to 1. When p is small the additional cost of the significance test is outweighed by the savings when the set is insignificant. This aggregation and testing is equivalent to grouping the X_i's into trees of height 1 and using the significance mapping procedure above.

Let $\mathbf{N}^{(1)}$ be the root of a tree of height 1 with K_1 variables X_i as children.We define p_1 to be the probability that $\mathbf{N}^{(1)}$ is significant, and we have $p_1 = 1-(1-p)^{K_1}$. The expected cost per coefficient of coding the coefficients in the tree $\mathbf{N}^{(1)}$ is $C_1 = \frac{1}{K_1} + p_1$. Our aggregation provides a cost savings over the coding coefficients individually when $p < 1 - \left(\frac{1}{K_1}\right)^{\frac{1}{K_1}}$.

Provided that p is sufficiently small, we can further reduce costs by iterating this process of aggregation. We form a height 2 tree with root $\mathbf{N}^{(2)}$ by grouping together K_2 of our height 1 trees and so on. Coding costs do not decrease indefinitely, however. The cost of the topmost significance check decreases at each iteration since it is amortized over more symbols, but this decrease in cost is offset by the more rapid decrease in the probability of insignificance.

Let p_j be the probability that the height j tree $\mathbf{N}^{(j)}$ is significant. We have $p_j =$

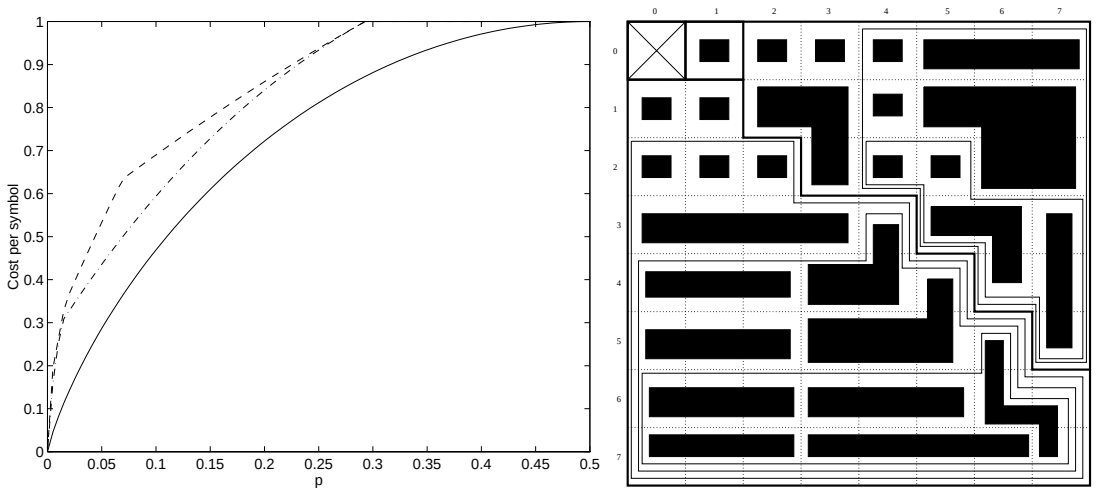

Figure 1: The figure on the left shows entropy vs. per-symbol cost of coding a collection of i.i.d. Bernoulli random variables using zerotrees. p is the probability of a 1. The solid curve is the entropy per symbol, and the dashed curve and the dashed-dotted curves are the zerotree costs with all nodes having 2 and 4 children respectively. All trees have maximal heights. The figure on the right shows an optimized partition of an 8x8 DCT block for the 512x512 Lena image.

$1-(1-p_{j-1})^{K_j}$, where $p_0 = p$. The cost per symbol for a height J tree is then

$$C_J = \left[\prod_{m=1}^{J} K_m\right]^{-1} \times \left[1 + \sum_{n=1}^{J}\left(p_n \prod_{l=J+1-n}^{J} K_l\right)\right] \tag{1}$$

Figure 1 compares the per-symbol cost of our hierarchical significance testing procedure to the entropy as a function of the probability of significance p. The singularities in the curve occur at the points at which we increase the number of levels in the hierarchical procedure. While the procedure does not achieve the limit except at $p = 0$ and $p = \frac{1}{2}$, it deviates by no more than 0.27 bits/symbol for $K = 2$ and 0.20 bits/symbol for $K = 4$ from $H(p)$ and provides a significant improvement over our crude initial 1 bit/symbol encoding at a very low computational cost. Thus, at the lowest level, zerotree quantization can be seen as providing a crude, low-complexity entropy coding for independent symbols.

2.4 Constrained Vector Quantization

Significance tree quantization offers greater potential for dependent random variables, since it allows exploitation of the coupling. For this case we will examine the performance of STQ when coupled with an ideal entropy coder. Consider a height 1 zerotree quantization of the variables X_1 and X_2 taking values in $\{0, 1\}$. Here we make no assumption that X_1 and X_2 are independent. Denote the root of this tree by $\mathbf{N}^{(1)}$ and let S be a random variable assuming the value 1 if the root node $\mathbf{N}^{(1)}$ is significant and 0 otherwise. The first step of STQ is to code the value of S. The expected bit cost of this encoding is $H(P(S = 1))$. If the root is significant, we then code the values of each of the X_i. For an optimal encoding we must condition the encoding of X_1 on S and X_2 on X_1 and S. Thus our expected cost is

$$C = H(S) + P(S = 1)\left[H(X_1|S = 1) + H(X_2|X_1, S = 1)\right] \tag{2}$$

An application of the chain rule shows that this cost is equal to $H(X_1, X_2)$, the joint entropy of X_1 and X_2.

The above 2-variable example shows that by using conditioning we can use STQ coupled with an arithmetic coder to code the pair X_1, X_2 at the Shannon limit provided we know the values of $P(S = 1)$, $P(X_1 = 1|S = 1)$, $P(X_2 = 1|X_1 = 0, S = 1)$, and $P(X_2 = 1|X_1 = 1, S = 1)$. In principle we can optimally code the significance of any number of X_i's in this way provided we employ appropriate conditioning. In practice, however, we do not have knowledge of all the conditional probabilities we require. Moreover, the number of context probabilities we need is exponential in the number of variables, and it becomes difficult to obtain accurate estimates of them. The SPIHT coder of Said and Pearlman [3] uses an adaptive conditioning scheme similar to the one described above for determining the significance of sets of 5 variables, but it is difficult to extend much beyond this without the estimation becoming impractical.

We can sidestep this problem of expanding contexts and still obtain an optimal encoding for an important class of random vectors. Moreover, this class is closely related to a simple model of the behavior of image transform coefficients, so our analysis provides an explanation of the good performance of STQ schemes in practice.

Consider a compound probability model for a pair of random variables X_1 and X_2. The model is a mixture of 2 different states, one deterministic and one random. The deterministic state occurs with probability α_1. In this state we have $X_1 = 0$ and $X_2 = 0$. The random state occurs with probability $1 - \alpha_1$. In this state the variables X_1 and X_2 are independent, identically distributed, and equal to 1 with probability p.

The probability that the magnitudes of two adjacent subband coefficients from an image exceed a given threshold can be approximated using the above model. Here the different states correspond to smooth and textured portions of an image. Transform coefficients are small in smooth regions of an image, which corresponds to the all-zero state. The coefficients in a textured image region are well approximated by i.i.d. random variables, which corresponds to the random state. In the absence of side information identifying particular coefficients as members of a smooth or textured region, we can model the state as a random variable.

As we saw above, we can optimally encode the 2-variable vector above using STQ with a small context model. The interesting thing is that we can maintain optimality for higher dimensional extensions of this model without having to increase the size of the context model.

Consider a vector of 4 random variables X_1, X_2, X_3, and X_4. We build a new compound model by joining together a pair of our above compound 2-vectors and adding a new state. In state 1, which occurs with probability α_2, the entire 4-vector is 0. In state 2, the pairs (X_1, X_2) and (X_3, X_4) are distributed independently according to our above compound model. We can continue this process and generate vectors of size 2^n for any n. These models are all obtained by taking sets of independent variables and coupling them by introducing states in which subsets of the variables are all zero. These models correspond to subband coefficients of images containing mixtures of textured and smooth regions of varying sizes. The model is not completely satisfactory, since the "smooth" and "textured" regions all are of dyadic length and begin and end on dyadic points, but it does capture some very important behavior of image subband coefficients.

We use a height n binary tree $\mathbf{N}^{(n)}$ to code the 2^n vector $\mathbf{X}^{(n)}$. The tree is formed by pairing the scalars X_{2k-1}, X_{2k} together to form the height 1 trees $\mathbf{N}_k^{(1)}$, then pairing the

trees $\mathbf{N}_{2k-1}^{(1)}, \mathbf{N}_{2k}^{(1)}$ together to form the tree $\mathbf{N}_k^{(2)}$ and so on. Let $S_k^{(j)}$ be a variable that is 1 if the node $\mathbf{N}_k^{(j)}$ is significant and 0 otherwise. We encode the tree in a recursive fashion as before, but now we use arithmetic coding with conditioning when coding the significance of a node. We code the significance of the root at a cost of $H(S_1^{(n)})$. If the root is insignificant, we are done. If not, we now code $S_1^{(n-1)}$ conditioned on the event $S_1^{(n)} = 1$, and we code $S_2^{(n-1)}$ conditioned on $S_1^{(n)} = 1$ and the value of $S_1^{(n-1)}$. We continue recursively coding all significant nodes, at each stage conditioning on the significance of the parent and of siblings. This local conditioning is all we need for optimal coding because of the way we have constructed the random variables X_i. The coupling between the variables is entirely due to their being zero together, and that coupling is taken care of by the hierarchical tests for significance. The exponentially growing contexts required for optimal coding in the general case collapse due to the constrained form of our model.

The excellent performance of the EZW and SPIHT wavelet coders can be explained in part by their ability to effectively exploit coefficient interdependencies of the sort described above. As we have seen, these models model important aspects of smooth and textured regions in images.

The performance of significance tree quantization is strongly dependent on the particular manner in which we group coefficients together to form zerotrees. In the next section we describe an algorithm for optimizing these trees for a given image or ensemble of images.

3 Optimized Significance Tree Selection

Test images compressed with the SPIHT coder [3] have PSNR's on the order of 1 dB higher than those reported for Shapiro's EZW coder [4]. The SPIHT coder makes minor improvements in EZW in a number of areas: it uses a slightly more effective filter set, it makes better use of context-based conditioning in its entropy coder, and it minimizes coefficient ordering effects. A large component of the PSNR improvement is due to an improvement in the construction of zerotrees. The SPIHT coder makes very effective use of the fact that when sets containing wavelet coefficients from many scales become significant, it is usually due to the coarsest scale coefficients in the set.

The zerotrees in both EZW and SPIHT were obtained from a number of assumptions about the behavior of wavelet coefficients in images. The zerotree structure in EZW is simple and elegant, but, as the SPIHT coder shows, it leaves considerable room for improvement. Is there more to be gained by improving our zerotree partitions? The algorithm below provides a means of answering this question.

An intriguing experiment by Xiong *et al.*[6] has shown that a SPIHT-style quantization is quite effective for the discrete cosine transform, yielding some of the best PSNR's in the literature for DCT-based coders. Although the zerotrees in [6] have been modified slightly to fit the structure of the DCT, they are still rather arbitrary, especially considering that the trees were originally designed for a wavelet transform. What are suitable zerotrees for the DCT? Our experiments below give optimized trees for particular images.

In the next section we introduce an algorithm for optimizing zerotrees for a given set of coefficients, whether they come from a wavelet transform or DCT block or from a raw image. We then describe a family of significance trees suitable for a DCT-based coder and we optimize our partitioning over this family. Our optimized zerotree encoder yields significant gains over that of [6].

3.1 Zerotree Optimization

Suppose we have a set of coefficients that we want to quantize using STQ. The set could be an 8×8 DCT block, for example, or a wavelet subtree of a fixed depth. Suppose also that we are given a training set consisting of M instances of this set. The discussion below assumes an additive error measure; in our experiments we use squared error. We further assume a fixed quantization scheme for coefficients determined to be significant. As a result, no matter what kind of tree structure we use to quantize the coefficients and the significance map, when we quantize for thresholds $T, \frac{1}{2}T, \ldots$ down to $2^{-n}T$, we obtain the same total error. The cost of quantizing significant coefficients is also independent of our zerotree structure. The cost of the significance map, however, depends strongly on the tree structure we use. The goal, then, is to find a tree structure for the significance mapping that minimizes the total cost of encoding all the members of the training set. This problem is the dual of the more commonly addressed problem of minimizing distortion with constrained cost.

Consider the cost of obtaining a significance map for a set of coefficients $\mathbf{\Sigma} = \{X_1, \ldots X_K\}$ by grouping them together into a tree with root node $\mathbf{N}(\mathbf{\Sigma})$. Our goal is to find the tree structure with root $\mathbf{N}(\mathbf{\Sigma})$ that minimizes the cost of determining the map for a fixed threshold T. Let $I(\mathbf{\Sigma},T)$ and $W(\mathbf{\Sigma},T)$ be the number of members in the training set for which $\mathbf{N}(\mathbf{\Sigma})$ is insignificant and newly significant, all with respect to T. These quantities are fixed by the coefficients in the set $\mathbf{\Sigma}$; changing the tree structure does not affect them.

We break the total cost of the significance map into two components. Let $R(\mathbf{\Sigma},T)$ be the cost of determining the significance of the root node $\mathbf{N}(\mathbf{\Sigma})$, and let $C(\mathbf{\Sigma},T)$ be the total cost of mapping the children of $\mathbf{N}(\mathbf{\Sigma})$. We consider the case of obtaining a map without entropy coding. The procedure below can, with minor modifications, be adapted for optimization with entropy coding.

In the absence of entropy coding we pay one bit for testing the significance of the node $\mathbf{N}(\mathbf{\Sigma})$ whenever it is insignificant or newly significant. The cost of the significance tests $R(\mathbf{\Sigma},T)$ is thus $I(\mathbf{\Sigma},T) + W(\mathbf{\Sigma},T)$. This root cost depends only on the coefficients contained in $\mathbf{\Sigma}$ and is not affected by the tree's structure.

The children of $\mathbf{N}(\mathbf{\Sigma})$ determine a partition of $\mathbf{\Sigma}$ into n subsets $\mathbf{\Sigma}_1, \ldots, \mathbf{\Sigma}_n$. Let $C(\mathbf{\Sigma}_j, T)$ be the total cost of mapping the children of the trees $\mathbf{N}(\mathbf{\Sigma}_1), \ldots, \mathbf{N}(\mathbf{\Sigma}_n)$ when these trees are not attached to the node $\mathbf{N}(\mathbf{\Sigma})$. Attaching the trees $\mathbf{N}(\mathbf{\Sigma}_1), \ldots, \mathbf{N}(\mathbf{\Sigma}_n)$ to the node $\mathbf{N}(\mathbf{\Sigma})$, results in a considerable cost savings when the nodes are all simultaneously insignificant. We need only pay for a single significance test at the root rather than for a significance test at each node. In all cases, we pay additional bits to determine the significance of $\mathbf{N}(\mathbf{\Sigma})$. This additional cost is reflected in the quantity $R(\mathbf{\Sigma},T)$. The total cost of mapping the children $C(\mathbf{\Sigma},T)$ is given by

$$C(\mathbf{\Sigma},T) = R(\mathbf{\Sigma},T) - nI(\mathbf{\Sigma},T) + \sum_{i=1}^{n} C(\mathbf{\Sigma}_j, T) \tag{3}$$

From (3) we see that our problem of minimizing the total cost of the significance map for the set $\mathbf{\Sigma}$ can be broken into a set of smaller minimization problems. We can therefore solve our minimization problem using dynamic programming. We minimize $C(\mathbf{\Sigma},T)$ by choosing partitions of $\mathbf{\Sigma}$ that minimize $C(\mathbf{\Sigma}_j, T)$. For example, we can make the partition significance tests $R(\mathbf{\Sigma}_j,T)$ small by grouping together coefficients that tend to be insignificant simultaneously. In the case of a wavelet transform significance map, this can be accomplished by grouping together fine-scale coefficients that are clustered together in space.

3.2 Families of Partitions

Finding an optimal significance tree for n coefficients requires that we solve subproblems involving all possible collections of $n-1$ coefficients, $n-2$ coefficients, and so on down to 1 coefficient. For even moderate values of n the number of possible partitions is enormous, so we cannot hope to find optimal cost partitions except for very small values of n. We consider instead the problem of optimizing trees over a given family, where the family has been chosen based on *a priori* knowledge about the given coefficients. Our search space, although constrained, is still quite large. For example, a wavelet analog of the search space described below would include both the EZW and SPIHT quantization schemes. While our optimization incorporates some assumptions about the form of our data, it uses far fewer than related significance tree schemes.

For our numerical experiments we seek optimized significance trees for 8×8 DCT blocks. DCT coefficients decay rapidly in frequency for most 8×8 blocks. As a result, high frequency coefficients will tend to be insignificant simultaneously. By grouping these coefficients together in our significance tree, we can in many cases indicate the insignificance of a large collection of high frequency coefficients by querying a single parent node. An important property of the family of partitions we will consider is that the partitions group high frequency coefficients together.

Some of the most important high frequency content of images comes from edges. DCT coefficients for isolated edges possess a well-defined structure. The coefficients possess a local maximum along a line extending from the DC origin. The angle of this line in frequency depends on the angle of the edge. Coefficient behavior is most easily described in polar coordinates. Coefficients fall off radially away from the frequency origin and as the angle moves away from the maximal angle. Because of this structure, DCT coefficients possessing similar angular coordinates will tend to be either significant or insignificant simultaneously. A second feature of the family of partitions we consider is that the partitions group coefficients with similar angular coordinates together.

We define a new coordinate system for each DCT block. Let $r = \max(x, y)$ and let $\theta = \arctan(\frac{y}{x})$ where x and y are frequency coordinates for the DCT coefficients each ranging from 0 to 7 and having origin at the DC value. We will restrict our attention to "rectangular" partitions in this polar coordinate system of the form $\{(r,\theta) : r_1 \le r \le r_2, \theta_1 \le \theta \le \theta_2\}$ which we will write as $[r_1, r_2, \theta_1, \theta_2]$.

4 Results

Our goal is to find an optimal significance tree for the AC coefficients of a DCT block, i.e. we seek to find an optimal tree for mapping the coefficients in the rectangle $[1, 7, 0, \frac{\pi}{2}]$. Finding this optimal tree requires that we solve a set of subproblems, each of which consists of finding an optimal tree for mapping coefficients in a rectangle $[r_1, r_2, \theta_1, \theta_2]$. We optimize the significance tree for $[r_1, r_2, \theta_1, \theta_2]$ by finding the lowest cost partition into children of the form $[r_1, r, \theta_1, \theta]$, $[r_1, r, \theta, \theta_2]$, $[r, r_2, \theta_1, \theta]$, and $[r, r_2, \theta, \theta_2]$. Each subproblem involves testing partitions for values of r between r_1 and r_2 and for values of θ between θ_1 and θ_2. This search is fast, since there are at most 8 distinct values of r and 16 distinct values of θ. We enlarge our search space slightly by also considering splits only in r, splits only in θ, and no split at all. We also allow merging parent and child nodes. The right hand side of figure 1 shows the result of our optimized partitioning for the DCT blocks in the 512×512 Lena image. The nested closed curves show how each group of coefficients is partitioned into subsets of coefficients. The blocks of black squares indicate coefficients at the lowest

level of the tree that are children of a single node.

The table below shows PSNR's at several rates for our optimized significance tree quantization for the 512x512 Lena and Barbara test images. The cost of coding the particular significance tree structure used, roughly 80 bytes, is included in the overall image cost. For comparison, we have included PSNR's for similar DCT-based schemes, including baseline JPEG[2], improved JPEG [5], and EZDCT without arithmetic coding[6]. We have optimized our significance trees for the case of no entropy coding, and the results shown in the table make no use of entropy coding. We note that arithmetic coding improves the EZDCT results by roughly 1 dB. A significance tree optimization procedure for the entropy coded case is currently under development.

Our significance tree quantization scheme (STQ) provides substantial performance improvements over baseline JPEG for the Barbara image, with improvements in PSNR ranging from 0.7 to 1.7 dB. STQ provides more modest improvements over EZDCT for Barbara at low bit rates, performing slightly worse at higher bit rates. Improved JPEG, however, always performs better than STQ on Barbara. For the Lena image, results are somewhat mixed. STQ provides modest gains over EZDCT at low rates and roughly equivalent performance at high rates. However, at low rates, JPEG and improved JPEG both yield better results. The reason for the relative decline in performance of these schemes at low rates is that the coding of DC values is very inefficient. In our experiments we have optimized significance trees only for the AC coefficients. DC coefficients have been coded using a significance tree based on [3] following [6]. At low bit rates this inefficiency in the DC coefficients is particularly pronounced and results in poor performance. The final column shows the result of performing a Haar transform on the DC values and coding the values using the coder of [3] with no entropy coding. This more effective coding of the DC values yields a substantial performance improvement at low rates.

Our optimization is fairly robust to significance tree mismatch. Switching the significance trees for Lena and Barbara leads to a relatively small loss of performance for Barbara (0.0-0.3 dB) and for Lena (0.0-0.5 dB). We are currently investigating significance trees optimized for ensembles of images of various classes, including motion compensated residual images.

	PSNR (dB)									
Rate	JPEG		Improved JPEG		EZDCT w/o ac		STQ		STQ + Haar	
(b/p)	Barb.	Lena	Barb.	Lena	Barb.	Lena	Barb.	Lena	Barb.	Lena
0.25	25.2	31.6	26.0	31.9	25.4	30.7	25.9	31.2	26.4	32.3
0.50	28.3	34.9	30.1	35.5	29.4	34.8	29.7	35.4	30.2	35.6
0.75	31.0	36.6	33.0	37.5	32.5	37.1	32.1	37.0	32.3	37.2
1.00	33.1	37.9	35.2	38.8	34.9	38.7	34.8	38.8	34.9	39.0

Table 1: PSNR's for baseline JPEG[2], improved JPEG[5], EZDCT with no arithmetic coding[6], and our significance tree quantization scheme.

5 Conclusion

Significance tree quantization provides an effective mechanism for exploiting coefficient interdependencies in both DCT-based and wavelet-based coders. This performance can be enhanced significantly by optimized significance tree design. Our optimization procedure

yields a fully embedded, low-complexity DCT-based coder that has significantly better performance than baseline JPEG.

6 Acknowledgments

This work has been supported in part by an NSF Mathematical Sciences Postdoctoral Research Fellowship and by ARPA, as administered by the AFOSR under contract DOD F4960-93-1-0567.

Zixiang Xiong has provided much useful feedback and invaluable assistance in the implementation and analysis of our algorithm. Discussions with Kannan Ramchandran originally motivated this work.

References

[1] T. Bell, J. G. Cleary, and I. H. Witten. *Text Compression.* Prentice Hall, Englewood Cliffs, NJ, 1990.

[2] W. B. Pennebaker and J. L. Mitchell. *JPEG Still Image Data Compression Standard.* Van Nostrand Reinhold, New York, 1992.

[3] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchichal trees. *IEEE Trans. Circuits and Systems for Video Technology*, 6(3):243–250, June 1996.

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

[5] S. Wu and A. Gersho. Rate-constrained picture adaptive quantization for JPEG baseline coders. In *Proc. ICASSP*, volume 5, pages 389–392, April 1993.

[6] Z. Xiong, O. Guleryuz, and M. T. Orchard. A DCT-based embedded image coder. *IEEE Signal Processing Letters*, Nov. 1996.