ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Signal Processing: *Image Communication* ▮ (▮▮▮▮) ▮▮▮–▮▮▮

SIGNAL PROCESSING: IMAGE COMMUNICATION

www.elsevier.com/locate/image

Group testing for image compression using alternative transforms ☆

Edwin S. Hong[a], Richard E. Ladner[b], Eve A. Riskin[c,*]

[a] *Computer and Software Systems, University of Washington, Tacoma, Box 358426, 1900 Commerce Street, Tacoma, WA 98402-3100, USA*

[b] *Department of Computer Science and Engineering, University of Washington, Box 352350, Seattle, WA 98195-2350, USA*

[c] *Department of Electrical Engineering, University of Washington, Box 352500, Seattle, WA 98195-2500, USA*

Received 8 January 2003; accepted 10 April 2003

Abstract

This paper extends the group testing for wavelets (IEEE Trans. Image Process. 11 (2002) 901) algorithm to code coefficients from the wavelet packet transform, the discrete cosine transform, and various lapped transforms. Group testing offers a noticeable improvement over zerotree coding techniques on these transforms; is inherently flexible; and can be adapted to different transforms with relative ease. The new algorithms are competitive with many recent state-of-the-art image coders that use the same transforms.

Keywords: Group testing; Image compression; Discrete wavelet transform; Wavelet packets; Block transform coding; Lapped transforms; Embedded coding

1. Introduction

Much of recent compression work has focused on efficient methods for encoding the transform coefficients of an image. Although the wavelet transform has received the most attention in recent years, alternative transforms such as wavelet packets and various block transforms have also been effectively applied to images. Different transforms have various characteristics such as ease of implementation or suitability for specific types of images, and hence may offer competitive advantages over the discrete wavelet transform in certain situations. In this paper, we extend the group testing for wavelets (GTW) algorithm [8] to alternative transforms including the wavelet packet transform, the discrete cosine transform (DCT), and several versions of lapped transforms [12,19]. The overall goal of this paper is to demonstrate the flexibility of group testing and its ability to be extended to different transforms in a straightforward manner.

As presented in [8], the group testing framework transforms an image and then encodes the resulting transform coefficients in a bit-plane order with

☆This work appeared in part at the 2001 Data Compression Conference and the 35th Asilomar Conference on Signals, Systems, and Computers. Research supported by NSF grant CCR-9732828, NSF Grant EIA-9973531, and NSF Grant CCR-0104800.

*Corresponding author.

E-mail addresses: edhong@u.washington.edu (E.S. Hong), ladner@cs.washington.edu (R.E. Ladner), riskin@ee.washington.edu (E.A. Riskin).

0923-5965/03/$ - see front matter © 2003 Published by Elsevier Science B.V.
doi:10.1016/S0923-5965(03)00050-X

many different adaptive group testers. For efficient compression, the coefficients are divided into classes whose coefficients have similar statistical characteristics. To apply this framework effectively on alternative transforms, new class definitions are needed. One main goal of this work is to discover the appropriate class definitions for each type of transform that will result in good performance.

Our work is partially motivated by previous work that applied zerotree coding (introduced in [17]) to these alternative transforms (see [11,13,15,20,24]). In particular, EZ-DCT [24] gave dramatic improvements over JPEG on both the Barbara and Lena images (see Table 3). The success achieved by EZ-DCT leads us to explore this area further.

The zerotree technique was motivated by the multi-resolution structure of the dyadic wavelet decomposition, where coefficients could be organized into trees formed across different subbands. Since there is a mismatch between the zerotree structure and the statistical characteristics of the coefficients generated from the alternative transforms we study, using zerotree coding on these coefficients may lead to some inefficiency in coding performance. Furthermore, there does not appear to be a natural way to define the parent–child relationships between the alternative transform coefficients, as there is in the dyadic wavelet decomposition.

As a generalization of zerotree coding, group testing is not hampered by the zerotree structure and can easily be adapted to more efficiently code these transform coefficients. Our results indicate that our group testing technique achieves better PSNR performance than previous zerotree coding techniques for a given transform.

Further, our new results show significant performance improvements over GTW on the Barbara image. On this image, the algorithm using the best lapped transform performed about 1.6 dB better than GTW at a wide range of bit-rates. Similarly, the wavelet packets version performed about 1.2 dB better than GTW. Other images also showed some improvement, although not quite as much. In addition, the algorithms also compare favorably to the JPEG 2000 standard.

This paper is organized as follows: Section 2 reviews the main elements of the framework that was used in the GTW algorithm. This includes a brief overview of group testing, image coding, and the GTW algorithm. Section 3 presents the group testing for wavelet packets (GTWP) algorithm, which includes a brief overview of wavelet packet image compression, the GTWP algorithm, and GTWP's rate-distortion performance. Section 4 presents the group testing for block transforms (GTBT) algorithm, including an overview of block transforms, and GTBT's performance results. We summarize our overall results in Section 5.

2. Group testing for image compression

2.1. Introduction

Group testing is a technique used for identifying a few significant items out of a large pool of items. In this framework, the significant items can be identified only through a series of *group tests*. A group test consists of picking a subset of items and testing them together. There are two possible outcomes of a group test on set K: either K is *insignificant* (meaning all items in K are insignificant), or K is *significant* (meaning there is at least one significant item in K). The goal is to minimize the number of group tests required to identify all the significant items. In this paradigm, the cost of testing any set of items for significance is the same as the cost of testing a single item.

As shown in [8], group testing can be viewed as a generalized form of zerotree coding, where the groups tested together do not have to be coefficients organized strictly into trees. The encoded output would simply be a series of bits representing the group test results; this is exactly like using bits to represent whether a tree of coefficients is significant in zerotree coding. Group testing for image compression replaces the zerotree coding process of a typical embedded zerotree coder with a technique based on group testing. Other methods besides zerotree coding for coding significant coefficients have also been previously studied. These include Andrew's hierarchical set partitioning [3] and Davis's significance tree quantization, which coded DCT coefficients from an 8×8 block in contiguous groups, and optimized the groups for performance [7].

2.2. Group testing framework overview

In our group testing framework, we follow the standard practice of applying a linear transform to the image data, and then coding the transform coefficients. The transform coefficients are coded in a bit-plane by bit-plane fashion, with each bit-plane coded by two passes: a significance pass that identifies newly significant coefficients in the current bit-plane, and a refinement pass that gives an additional bit of precision to already significant coefficients.

The significance pass uses an adaptive form of group testing based on *group iterations* (described in Section 2.3.1). Since this adaptive method is known to work well on i.i.d. sources, we try to ensure that the coefficients we code are approximately i.i.d. We accomplish this by dividing the coefficients into *classes*, where each class is coded by a different adaptive group tester. The classes are designed so that coefficients within one class are well approximated by an i.i.d. source. Note that dividing the coefficients into classes is similar to choosing a different methods of coding coefficients based on context.

Since the statistical characteristics of the transform coefficients depend on the transform used, the classes should be designed separately for each transform. In [8], the GTW classes were designed for the dyadic wavelet decomposition of an image. In this work, we design new classes for the alternative transforms that we use. We present several different definitions of classes in Sections 3.4 and 4.5.1.

For the purposes of obtaining good embedded performance, we code the classes in order of the probability of significance of their coefficients. Classes with coefficients that have a higher probability of being significant are encoded first. Since the probability of significance of the coefficients in any class depends on the class definition, we order the classes based on the class definition.

2.3. Some significance pass details

We first describe group iterations, the method by which our significance pass is encoded. We then describe our adaptive group testing strategy. Finally, we then end this subsection with a description of how the group testing framework encodes the different classes using adaptive group testing. This section only presents an overview of our significance pass; for full details, see [8].

2.3.1. Group iterations

A group iteration is a simple procedure that is given a set K of k items, and uses group tests to identify up to 1 significant item, and up to k insignificant items. At the end of a group iteration, there may be some unidentified items in K that must be tested in a future group iteration. If the set K contains a significant item, the group iteration will use $\lceil \log_2 k \rceil + 1$ group tests in a recursive, binary search-like process to identify one significant item; otherwise it will use exactly one group test to identify set K as containing only insignificant items.

2.3.2. Adaptive group testing

We adaptively pick the group iteration size k depending upon the statistical characteristics of the items being encoded. We start out initially in a doubling phase, with group iteration size 1, and double the size of each successive group iteration as long as no significant items have yet been found. Once a significant item has been found, we move to the steady-state estimation phase, where we choose a group iteration size that results in optimal coding performance based on our estimate of the probability of significance. Our estimate is calculated as the percentage of significant items seen so far.

2.3.3. Significance pass algorithm

As previously described, our method divides the coefficients of one bit-plane into classes, and uses the previously described adaptive group testing technique to code each class. Given the class ordering and the definition of classes, the algorithm for encoding the significance pass is conceptually very simple:

Pick the first class (according to the class ordering) that contains enough coefficients. Then perform a group iteration of size k on that class, where k is chosen according to the statistics in the adaptive group tester for that class. Then update the coefficients as necessary with the information learned from the group tests (coefficients could change classes at this point). Finally, repeat this entire procedure until all coefficients are coded.

3. Group testing for wavelet packets

3.1. Wavelet packets background

As described in [23], wavelet packets are a generalization of the standard dyadic wavelet decomposition of a signal. The standard dyadic wavelet transform decomposes the signal by applying a series of successive filters to the lowest frequency subband. Wavelet packets are a generalization of this where the successive filters can be applied to any subband of any orientation, not just the lowest frequency LL subband. Any one particular choice of subbands to decompose is known as a basis; the choice of exactly which basis to use depends on the characteristics of the input. Fig. 1 shows the subbands after transforming an image with the wavelet packet transform using one particular basis.

A basis that adapts well to the input signal can be chosen via Coifman and Wickerhauser's entropy-based technique [6] or by Ramchandran and Vetterli's rate-distortion optimization technique [16]. These methods work by fully decomposing all subbands to a predefined maximum depth, thus forming a decomposition tree where each decomposed subband is represented in the tree by a parent node with four child nodes. Then the best basis is found by pruning this decomposition tree in a recursive bottom-up fashion. The entropy-based technique prunes the tree to minimize the overall estimated entropy of the wavelet packet structure. The rate-distortion method is given a particular target bit rate for the image and prunes the tree to minimize the distortion of the image.

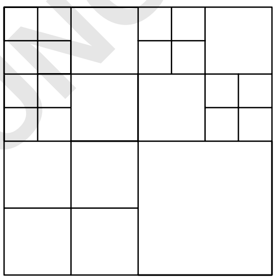

Fig. 1. Sample subbands of a wavelet packet-transformed image.

Xiong et al. [25] first explored the combination of a wavelet packet decomposition of an image with the space-frequency quantization (SFQ) coder, a coder that uses zerotree quantization techniques. The difficulty in applying zerotree quantization to wavelet packets is that it is no longer clear how to define the parent–child relationships in the trees. As noted by Rajpoot et al. [15], there is a *parenting conflict*, where some child coefficients could have multiple parents. This problem has typically been solved by limiting the space of possible wavelet packet decompositions so that no parenting conflict occurs, or by assigning the parent–child relationships in a somewhat ad hoc manner (see [11,15,25]). We believe that it is preferable that the coder not constrain the wavelet decomposition.

3.2. Group testing for wavelet packets

We propose a new coder, group testing for wavelet packets (GTWP), that applies our group testing framework to the wavelet packet transform. The first step is to find the best basis for the input image, and encode the structure of this basis in the first bits of our compressed image. Then we define the GTWP classes based on the characteristics of the wavelet packet decomposition of the image, so that the classes are encoded efficiently. Along with the class definition, we also specify the order in which we will code the classes. Once both the GTWP classes and the ordering between them are defined, then we can code each class with a different group tester, and proceed as described in the group testing framework for image compression.

We first describe how we choose the best basis and encode it; then we describe a method for defining the GTWP classes with their associated orderings.

3.3. Best basis

We investigated using both the entropy-based technique and the rate-distortion technique for computing the best wavelet packet basis. For the entropy-based technique, we explored many different metrics for calculating the entropy of a particular subband. Let v_i represent the value of the coefficients of a subband. Then the entropy metrics we tried are as follows:

- log energy metric: $\sum_i \ln(v_i^2)$,
- Shannon metric (used in [6]): $\sum_i v_i^2 \ln(v_i^2)$,
- L1-norm metric: $\sum_i |v_i|$,
- threshold metric: given a threshold value T, calculate $|\{v_i\colon |v_i| > T\}|$,
- first-order entropy metric (used in [11]): given a quantization step size Q, divide the coefficients into quantization bins, and estimate the probability p_i of a bin occurring by $p_i = n/t$, where n is the number of coefficients in that bin, and t is the total number of coefficients. Calculate $-\sum p_i \log_2(p_i)$.

We also tried the rate-distortion optimization technique, optimizing for a wide variety of bit-rates for various different possible scalar quantizers. Note that this technique is not well suited to our problem because it forces us to pick artificial parameters, namely, the final bit-rate for which to optimize and the quantizer step sizes to consider. Since GTWP is an embedded coder, the final bit-rate we choose for the purpose of obtaining the best basis does not correspond to the actual final bit-rate to which we encode the image. Furthermore, since GTWP codes the transform coefficients bit-plane by bit-plane, it cannot choose to code a subband with a particular quantizer step size; the step size it ends up using may not have any relation to the quantizer step size parameters that we chose to run the rate-distortion optimization technique.

It is interesting to note that for the Barbara image, the optimal calculated quantizer step size for all the subbands under the rate-distortion technique differed from each other by no more than a factor of 2. In the bit-plane encoding technique, if we stop coding in the middle of a bit-plane, then the coefficients that have not yet been coded in the current bit-plane are quantized with a step size of two times the step size of those coefficients that have been coded. This suggests that GTWP's bit-plane encoding technique may be a good approximation to the quantization step sizes that the rate-distortion optimization best basis produces.

The log energy, Shannon, and L1-norm metrics are the simplest in that they do not require additional parameters (such as threshold value or quantization step size) to compute. The top performers for our algorithm are the log energy metric and the rate-distortion optimization metric. Seeing that the log-energy metric was simpler and did not require selecting artificial parameters, we used it exclusively. As an example, we show the best basis chosen by the log-energy metric on the Barbara image in Fig. 2. For simplicity, we show only five levels of decomposition even though our algorithm uses a maximum of six levels. Notice how the wavelet packet decomposition reflects the large number of diagonal and vertical edges in the Barbara image, and how much it differs from the standard DWT decomposition.

To encode the decomposition tree, we simply perform a depth-first traversal of the tree, and encode a 1 when that particular node is split into children, and a 0 when the node is a leaf.

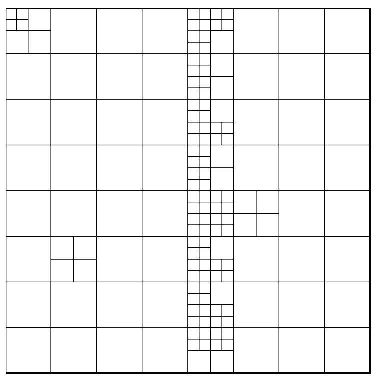

Fig. 2. Illustration of the best basis for the Barbara image.

3.4. GTWP classes

For simplicity and to allow convenient comparisons against JPEG 2000, the GTWP classes are based on the contexts in Taubman's EBCOT coder [18] (also found in the JPEG 2000 coder). In this class definition, there are only two characteristics that define the classes: the orientation type and the neighborhood significance label. We also tried a variant of GTWP based on classes that were very similar to those used in GTW, and obtained similar results to the JPEG 2000-based classes.

3.4.1. Orientation type

The orientation type of a subband is either LH, HH, or HL as illustrated in Fig. 3. A wavelet packet subband is of orientation type LH if it is contained in a *wavelet* subband which is the horizontal low pass and the vertical high pass component. In addition, the lowest-level subband (the LL subband) is considered to have orientation-type LH. Similarly, a wavelet packet subband is of orientation HH (HL) if it contained in a *wavelet* subband which is the horizontal high pass and the vertical high (low) pass component.

3.4.2. Neighborhood significance label

Let h, v, and d represent the number of significant neighbors that a coefficient has which are adjacent to it horizontally, vertically, and diagonally, respectively. Thus, h and v both have a value of up to 2, whereas d has a maximum value of 4. The neighborhood significance label is assigned according to Table 1. Note that the labeling is dependent on the orientation type of the

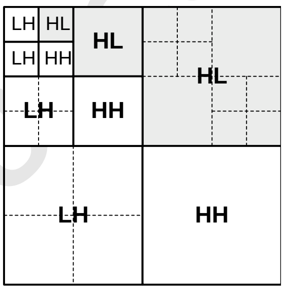

Fig. 3. Illustration of orientation type of a wavelet-packet transformed image. All shaded coefficients have orientation-type HL.

Table 1
Neighborhood significance label

Assigned label	LH subband			HL subband			HH subband	
	h	v	d	h	v	d	d	$h+v$
0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0	1
2	0	0	$\geqslant 2$	0	0	$\geqslant 2$	0	$\geqslant 2$
3	0	1	$\geqslant 0$	1	0	$\geqslant 0$	1	0
4	0	2	$\geqslant 0$	2	0	$\geqslant 0$	1	1
5	1	0	0	0	1	0	1	$\geqslant 1$
6	1	0	$\geqslant 1$	0	1	$\geqslant 1$	2	0
7	1	$\geqslant 1$	$\geqslant 0$	$\geqslant 1$	1	$\geqslant 0$	2	$\geqslant 0$
8	2	$\geqslant 0$	$\geqslant 0$	$\geqslant 0$	2	$\geqslant 0$	$\geqslant 2$	$\geqslant 0$

coefficient. This label is taken from the context classifier in the EBCOT coder.

With three orientation types and nine significant neighbor labels for each orientation type, there are a total of 27 classes. The classes are dynamically ordered according to the group iteration size. Classes with smaller group iteration size are coded first, since they are more likely to be significant. Ties are broken arbitrarily.

In our investigations on how to organize the coefficients into classes we discovered that the depth of the wavelet packet decomposition was not a good predictor of the magnitude of the coefficients. Since the decision of whether to decompose a subband is determined locally, the magnitudes of coefficients at the same depth can vary widely. In particular, the magnitudes of the lowest-frequency subband are usually much higher than those of all other subbands.

3.5. Results

Here we present our results on a set of standard eight-bit monochrome images: the 512×512 images Barbara et al. (available from [21]); and a 768×768 fingerprint image from the FBI's fingerprint compression standard [5]. We present results for several different algorithms, including GTW, GTWP, JPEG 2000 and SFQ-WP [26]. All algorithms use the Daubechies $\frac{9}{7}$-tap filters [4]. JPEG 2000 results were produced with a beta version of a codec [2] for the JPEG 2000 image

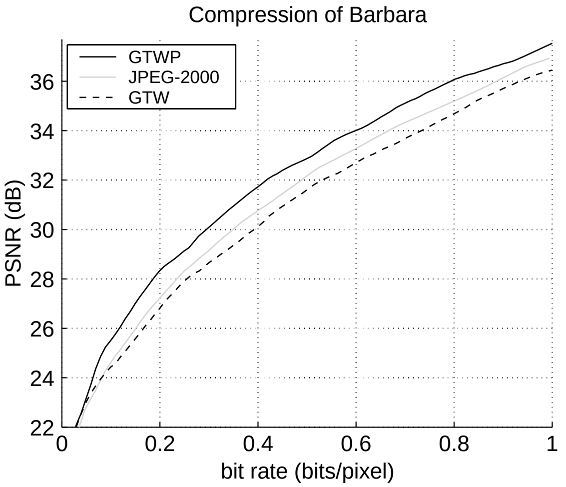

Fig. 4. Comparing performance of GTW, GTWP, and JPEG 2000.

compression standard. SFQ-WP represents the practical version of Xiong et al.'s SFQ algorithm applied with wavelet packets; results are taken from [26]. To our knowledge, SFQ-WP is the current state-of-the-art method for image compression with wavelet packets.

Fig. 4 compares the PSNR curves for GTW, GTWP, and JPEG 2000 on the Barbara image. As can be seen, using wavelet packets increases the PSNR by about 1.2 dB over GTW.

Table 2 lists PSNR results for all four images on the different algorithms. It shows that the amount of improvement for using wavelet packets instead of the dyadic wavelet decomposition is highly dependent on the type of image. Some images (like Barbara) benefit significantly from the wavelet packet decomposition; some images (like Goldhill) benefit slightly; and some (like Lena) do not benefit at all. In fact, the best wavelet packet basis for the Lena image was calculated to be the standard dyadic wavelet decomposition with one additional decomposition of a highest-frequency subband. Thus, as expected, the results for GTWP on Lena are roughly the same as that for GTW. The slight performance differences are due mostly to differing significant neighbor metrics. In fact, it appears that the significance of a coefficient depends almost entirely on its eight immediately adjacent neighbors, and very little on the parents and other neighbors in different subbands. This agrees with the findings in [18].

Table 2
Comparing PSNR results of various algorithms

Image	Algorithm	Rate (bits/pixel) 0.1	0.25	0.5	1.0
Barbara	GTW	24.37	27.87	31.59	36.47
	ΔJPEG 2000	+0.19	+0.46	+0.50	+0.46
	ΔGTWP	**+1.15**	+1.27	+1.28	+1.07
	ΔSFQ-WP	NA	**+1.38**	**+1.53**	**+1.22**
Lena	GTW	**30.06**	34.13	37.27	40.51
	ΔJPEG2000	−0.27	−0.12	−0.13	−0.46
	ΔGTWP	−0.02	+0.09	−0.01	−0.09
	ΔSFQ-WP	NA	**+0.22**	**+0.13**	**+0.04**
Goldhill	GTW	27.76	30.46	33.10	36.47
	ΔJPEG2000	−0.05	+0.04	+0.05	−0.07
	ΔGTWP	**+0.01**	**+0.09**	**+0.17**	**+0.13**
Fingerprint	GTW	28.35	32.80	36.06	39.98
	ΔJPEG2000	+0.24	+0.17	+0.10	+0.16
	ΔGTWP	**+0.45**	**+0.31**	**+0.70**	**+1.38**

Best results in boldface. GTW is used as the baseline, so that for algorithm **A**, Δ**A** = **A** − GTW.

If we compare our results with the published results of previous zerotree coding techniques on wavelet packets, we see that we outperform Rajpoot et al.'s technique by over 1.0 dB on the fingerprint image, and we outperform Khalil et al.'s technique by about 0.3 dB on the Barbara image.

As can be seen in Table 2, GTWP's performance is always better than JPEG 2000. Furthermore, GTWP's performance is not too far from that of SFQ-WP for the Barbara and Lena images. Although GTWP is worse, GTWP is an embedded coder, while SFQ-WP is not. Furthermore, GTWP, unlike SFQ, neither uses arithmetic coding nor performs rate-distortion optimization.

4. Group testing for block transforms

In this section, we show the results of applying our group testing framework to some standard block transforms. We first overview the use of block transforms for image compression. We then define the classes that we use for the block transforms, and conclude with a discussion of our results.

4.1. Block transform overview

4.1.1. Block transform background

When applying standard block transforms such as the DCT to images, the input pixels are divided into $M \times M$ blocks, and each block is separately transformed into an output block of size $M \times M$. The coefficient at position $(0, 0)$ in each output block is known as the *DC coefficient*, and all other coefficients are known as *AC coefficients*. Note that the DC coefficient of a particular block represents the average of the pixel values in the block.

A lapped transform [12] is a generalization of the standard block transform where the input is divided into overlapping blocks of length L, with each block transformed into an output block of size M; we call this an $M \times L$ lapped transform. An $M \times L$ lapped transform can be computed by multiplying the input row vector of length L with an $L \times M$ size matrix representing the transform, resulting in a output block of length M. In a typical example where $L = 2M$, each input data point is used in two adjacent output blocks. In this case, the inverse transform to recover one original block of M input data points is computed by taking two adjacent output blocks of coefficients ($2M$ coefficients total) and multiplying it with another $2M \times M$ matrix representing the inverse transform. In the two-dimensional case, we can view a lapped transform as mapping overlapping input blocks of size $L \times L$ into output blocks of size $M \times M$.

Unlike non-lapped block transforms, lapped transforms can take correlation between adjacent blocks into account; this makes it more efficient at decorrelating signals. Lapped transforms can also reduce blocking artifacts because their basis functions decay smoothly to near zero at the boundaries. Both lapped orthogonal transforms (LOT) and lapped biorthogonal transforms (LBT) have been studied. LBTs have more degrees of freedom than LOTs since the biorthogonality condition is weaker than the orthogonality condition. Aase and Ramstad [1] have shown that these extra degrees of freedom can be used to design better lapped transforms for image coding.

4.2. Organization into subbands

The block-transform coefficients of an $LM \times LM$ image are typically stored in a block-by-block fashion, so that the output of a block transform that uses $M \times M$ blocks consists of an $L \times L$ grid of $M \times M$ blocks, where each block represents an $M \times M$ block of the original input image. However, we can conceptually reorder the transform coefficients into a grid of $M \times M$ subbands, each of size $L \times L$. This reordering puts all the DC coefficients into one subband, ordered so that the DC coefficient in block (x, y) is at position (x, y) in the DC subband. Similarly, there will be a separate subband for each AC coefficient; subband (i, j) will contain AC coefficients from position (i, j) within their block, ordered so that the AC coefficient at position (i, j) in block (x, y) will be located at position (x, y) in subband (i, j). Fig. 5 illustrates this reorganization when $M = 4$ and $L = 6$.

In this reorganized picture, each of the M^2 subbands represents the entire original image at a different frequency decomposition. Note that with this organization, these subbands are similar to the subbands from a dyadic wavelet decomposition in that coefficients in a subband represent the same frequency decomposition of an image over differing spatial locations. Furthermore, the upper-left block of DC coefficients (see Fig. 5) represents a postage-stamp size overview of the entire image, much like how the lowest-frequency subband in a dyadic wavelet decomposition gives an overview of the image. The principal difference between the dyadic wavelet decomposition and this reorganized block transform picture is that all the subbands from block transforms are the same size, whereas in the wavelet transform, the subbands' sizes decrease by a factor of 2 with every additional level of the DWT performed. In other words, block transforms offer a *uniform-band* frequency partitioning of the input, in contrast to the *octave-band* frequency partitioning of the wavelet transform (see [20]).

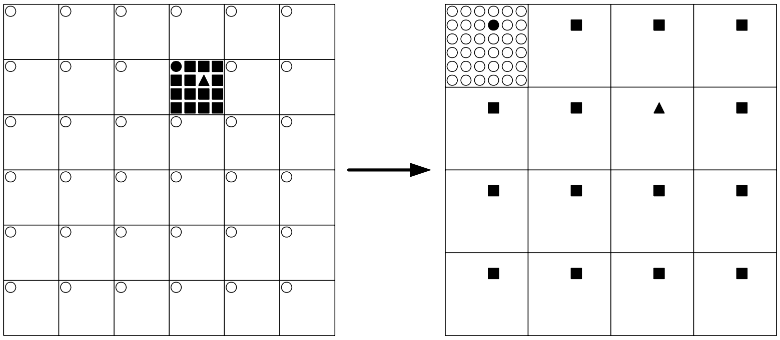

Fig. 5. Block transform coefficients on the left are reorganized into subbands on the right. The DC coefficients are represented as circles and end up together in one subband; black coefficients from one block are scattered out to all subbands.

For the DCT transform, the DC coefficient of an output block represents an average of the 8×8 input block. Since adjacent image blocks often are similar, adjacent coefficients in the DC subband will be correlated. For the lapped transforms, each traditional $M \times M$ output block is computed from an $L \times L$ input block of the original image. Most of the energy in the DC coefficient of the lapped transform is from the average of the entire $L \times L$ block. Since blocks are overlapping, some image pixels are used in more than one average and contribute their energy to adjacent coefficients in the DC subband.

4.3. Relation to the wavelet transform

With the subband organization, it becomes clear that we can also perform several levels of block transforms by recursively reapplying the block transform to the DC subband. We use the term *hierarchical block transform* to refer to any block transform scheme that decorrelates its DC subband by applying another transform. Note that hierarchical block transforms are similar to the levels of the DWT in a dyadic wavelet decomposition. Since the DC subband represents a small low-resolution overview of the entire image, we expect there to be significant correlation in the DC subband. Hierarchically reapplying a block transform to the DC subband should decorrelate it further and enable better compression performance. We could continue to perform levels of the block transform as long as the lowest-frequency DC subband is not too small. Note that after every block transform step, we always reorganize the transform coefficients so that a DC subband is always present. Also note that in principle, any transform could be used to decorrelate the DC subbands; in addition to the lapped transforms and the DCT, even a DWT could be used to decorrelate the DC subband.

Another relationship between lapped transforms and the DWT is that a lapped transform can be thought of as a generalization of one level of the DWT. Recall that the output coefficients of a wavelet transform can be computed via convolution. For a k-tap wavelet transform, any one output coefficient depends on at most k consecutive input coefficients. Thus, an $M \times (M + k)$ lapped transform can use the overlap of k data points on the input to compute the convolution of the input with the wavelet filter coefficients as would be done by the DWT. In other words, the DWT can be implemented as a lapped transform. Furthermore, hierarchical lapped transforms can completely implement the DWTs that use many levels. In its full generality, hierarchical block transforms have the potential to perform better than the DWT.

4.4. Previous block transform zerotree coders

The most widespread image compression format using DCT is the standard JPEG [22] format. It uses 8×8 DCT blocks. Xiong et al.'s embedded zerotree DCT algorithm (EZ-DCT) [24] applied the zerotree technique to the DCT-transformed coefficients of an image. Although the coefficients of a DCT transform are not naturally tree-structured, this coder showed that by imposing a somewhat arbitrary tree structure on the coefficients, reasonable performance could be achieved, certainly better than JPEG.

Malvar applied the zerotree technique to lapped transform coefficients [13,24]. He basically used the same method as EZ-DCT, but replaced the DCT transform with lapped transforms. He defined an 8×16 LOT transform as well as a 8×16 LBT transform that were optimized for both image compression efficiency and low computational requirements. We use EZ-LOT (EZ-LBT) to refer to Xiong et al.'s embedded zerotree technique when applied to Malvar's fast version of the LOT (LBT) transforms.

Tran et al. [13,20,24] focused on designing the best lapped transforms for image compression, and did not consider the speed of computation to be a crucial factor. They designed several lapped transforms, including the 8×16 generalized LBT (GLBT). This transform was optimized solely for good coding performance on images. They used a hierarchical coder that performed additional transforms on the DC components and a zerotree coding scheme that was very similar to the EZ-DCT scheme. We use EZ-GLBT to refer to Tran et al.'s hierarchical zerotree coder when applied to the GLBT transform.

4.5. Group testing for block transforms

In this section, we introduce our group testing for block transforms algorithm (GTBT) [10], which applies the group testing framework to code block transform coefficients. In describing our technique, we simply need to define the GTBT classes, and the ordering in which we code the GTBT classes. Our group testing framework then uses these definitions to code the transform coefficients in the significant pass. We also try both hierarchical and non-hierarchical versions of each of these block transforms. We now proceed to present GTBT classes, followed by the ordering we use when we code these classes.

4.5.1. GTBT classes

Similar to the definition of GTW classes, our GTBT classes have two different defining characteristics: the subband level and the significant neighbor metric.

In the definition of classes, we will exclusively use the subband view of the block-transform coefficients, as shown in Fig. 5. Thus, the transform coefficients have M^2 subbands, where M is the output block size of the transform.

4.5.1.1. Subband level. It is well known that for any particular block, the energy in the AC coefficients decreases as you move away from the DC coefficient. This means that coefficients in subbands closer to the DC coefficient are more likely to be significant than those in subbands farther away from the AC coefficient. To model this behavior, we classify subband (i,j) according to its distance from the DC coefficient at position $(0,0)$.

The DC subband is in subband level 0. Subband level 1 contains those subbands (i,j) which satisfy $i+j=1$ or $i+j=2$. Subband level 2 contains those subbands (i,j) which satisfy $2<i+j\leqslant 5$. Subband level 3 contains those subbands (i,j) which satisfy $5<i+j\leqslant M$, where M is the output block size, and subband level 4 contains those subbands whose position (i,j) satisfy $M<i+j$. Fig. 6 shows the five subband levels. This method of defining the subband levels was made based on the two competing goals: preventing context dilution (by having a small number of subband levels) and making the probability of significance of coefficients in one subband level about the same (by having many subband levels).

For hierarchical transforms that have two levels, we basically double the number of subband levels; one set of levels contain subbands from the first transform, and the other set contains the subbands from the second-level transform of the DC subband of the first transform. Since a two-level hierarchical transform does not have any DC coefficients from the first level, there are actually only nine subband levels in a two-level hierarchical transform. The subbands from the second level are considered more important than the subbands from the first level of the transform.

4.5.1.2. Significant neighbor metric. As expected, adjacent coefficients in a subband show statistical dependencies on each other. Here, we consider the neighbors of a coefficient to only be the eight coefficients in the same subband that surround a coefficient. Thus, we define four values in the significant neighbor metric: `0`, `1`, `2`, and `3+`, depending on whether 0, 1, 2, or 3–8 neighbors are significant. Note that this definition is a simplification of the original GTW scheme; this definition omits the parent–child relationships present in the GTW metric.

For non-hierarchical transforms, there are five subband levels and four significant neighbor types, resulting in 20 classes total. For two-level hierarchical transforms, there are nine subband levels, resulting in 36 classes total. These classes are once more ordered with significant neighbor metric considered more important than subband level.

4.6. Results

Here we again present our results on the Barbara, Goldhill, Lena, and the fingerprint images. We present results for our GTBT on several different transforms: the 8×8 DCT; the 8×16 type-I fast LOT with angles $(\theta_0, \theta_1, \theta_2) = (0.145\pi, 0.17\pi, 0.16\pi)$ [12]; the 8×16 LBT from [24]; the 8×16 GLBT from [20]. For convenience, we refer to GTBT using the DCT, LOT, LBT, or GLBT transforms as GT-DCT, GT-LOT, GT-LBT, and GT-GLBT, respectively.

First, we compare PSNR results of a non-hierarchical version of GT-DCT with Xiong et al.'s EZ-DCT scheme (results are taken from [24]), and

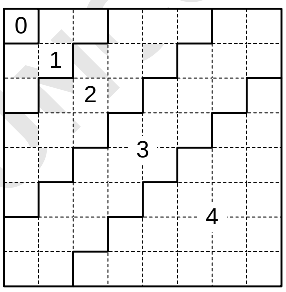

Fig. 6. The subband levels of a block-transformed image in GTBT. Solid lines separate subband levels, dotted lines separate subbands.

Table 3
Comparing PSNR results of coders using the non-hierarchical 8×8 DCT transform

		Rate (bits/pixel)			
Image	Algorithm	0.25	0.5	0.75	1.0
Barbara	GT-DCT	**27.12**	**31.02**	**33.97**	**36.11**
	EZ-DCT	26.83	30.82	33.70	36.10
	JPEG	25.1	28.49	31.28	33.26
Lena	GT-DCT	**32.30**	35.87	38.04	39.50
	EZ-DCT	32.25	**36.00**	**38.06**	**39.62**
	JPEG	31.67	34.9	36.67	37.94

In all cases, the best result is in boldface.

with standard JPEG. As Table 3 shows, GT-DCT performs somewhat better than EZ-DCT on Barbara, and about the same on Lena. The 0.3 dB difference on Barbara illustrates the improvement from using group testing instead of zerotree coding. Further, both GT-DCT and EZ-DCT significantly outperform standard JPEG. As expected, bit-plane coding works better than quantization followed by entropy coding.

Next, we show our results for hierarchical versions of the four different transforms we considered. For a given first-level block transform, we tried using many different transforms at the second level of the hierarchy. However, we found that the rate-distortion performance difference between different transforms at the second level was insignificant. Using any second level transform, however, did show a noticeable improvement over the non-hierarchical version. Given this fact, we only present results for hierarchical transforms that apply the same transform for the first and second level of the hierarchy. PSNR results are shown in Table 4.

Note that the transforms we use all produce 8×8 output blocks of coefficients, and 64 total subbands. Thus, after a two-level hierarchy on 512×512 images, the size of the second level DC subband is 8×8; this is small enough that applying another level of transform would not help.

Table 4
Comparing PSNR results of various algorithms

Image	Algorithm	Rate (bits/pixel)			
		0.1	0.25	0.5	1.0
Barbara	GTW	24.37	27.87	31.59	36.47
	ΔGT-DCT	−0.20	−0.29	−0.32	−0.16
	ΔGT-LOT	+0.85	+1.17	+1.20	+1.05
	ΔGT-LBT	+1.15	+1.49	+1.61	+1.39
	ΔGT-GLBT	**+1.27**	**+1.58**	**+1.65**	**+1.54**
Lena	GTW	**30.06**	**34.13**	**37.27**	**40.51**
	ΔGT-DCT	−1.69	−1.58	−1.22	−0.87
	ΔGT-LOT	−0.96	−0.95	−0.79	−0.63
	ΔGT-LBT	−0.46	−0.46	−0.38	−0.39
	ΔGT-GLBT	−0.45	−0.32	−0.20	−0.31
Goldhill	GTW	27.76	30.46	33.10	36.47
	ΔGT-DCT	−0.63	−0.54	−0.55	−0.47
	ΔGT-LOT	−0.14	0.00	−0.07	−0.07
	ΔGT-LBT	**+0.05**	+0.14	**+0.11**	+0.09
	ΔGT-GLBT	**+0.05**	**+0.15**	+0.10	**+0.14**
Fingerprint	GTW	28.35	32.80	36.06	39.98
	ΔGT-DCT	−1.15	−1.05	−0.44	+0.30
	ΔGT-LOT	−0.13	−0.23	+0.20	+0.93
	ΔGT-LBT	+0.12	+0.09	+0.47	+1.11
	ΔGT-GLBT	**+0.16**	**+0.23**	**+0.71**	**+1.41**

Best results in boldface. GTW is used as the baseline, so that for algorithm **A**, Δ**A** = **A** − GTW.

In general, coding performance improves as we proceed in order from DCT to LOT to LBT to GLBT. This is expected, because each transform is of higher quality than the previous one. LOT is better than DCT because it is lapped; LBT is a general form of the LOT; and the GLBT throws away the fast-computability characteristic of the LBT.

It also appears that the lapped transform methods perform much better than the standard GTW on only Barbara and fingerprint, the two images containing the most edges. These edges lead to more energy in the high-frequency coefficients. This suggests that the lapped transforms are better than the dyadic wavelet decomposition at decorrelating high-frequency content. One possible reason for this behavior is that the lapped transforms offer a uniform-band frequency partitioning of the input, compared with the octave-band partitioning found in the wavelet transform. As mentioned in [20], the finer frequency partitioning increases the frequency resolution and can often generate more insignificant coefficients.

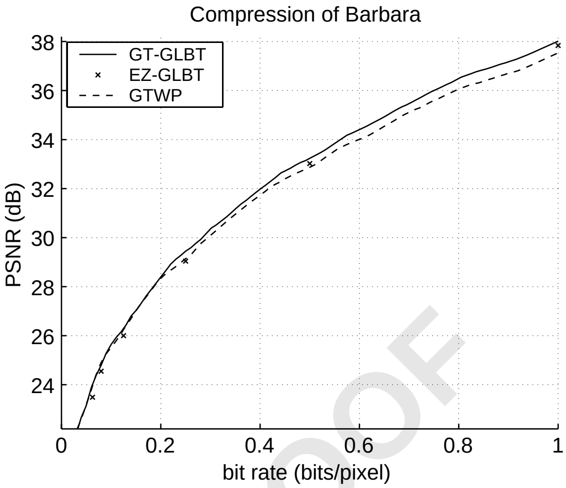

Fig. 7. Comparing performance of coders using the GLBT transform and GTWP.

Surprisingly, the GT-GLBT outperforms GTWP by about 0.30 dB (see Fig. 7) on the Barbara image. This is especially significant because GTWP already outperforms GTW on this image by about 1.3 dB.

As expected, GT-GLBT also shows improvement over Tran et al.'s Embedded Zerotree technique applied to the GLBT (EZ-GLBT). Improvements of up to 0.45 dB were observed on the Barbara image, as illustrated in the PSNR curves of Fig. 7. EZ-GLBT results are taken from [20]. This figure shows the performance gain of using group testing instead of zerotree coding in image compression.

5. Conclusion

In conclusion, we have shown that group testing is a general and flexible method that can be easily adapted to encode transform coefficients generated from wavelet packets as well as many block transforms. In most cases, designing classes based on a coefficient's number of significant neighbors leads to the best compression performance. We have shown that GTWP has good PSNR performance, in that it outperforms previously published zerotree coders that used wavelet packets. We have also shown that group testing of both DCT and GLBT transform coefficients improves upon zerotree coding of these coefficients by an average of 0.3 dB, over a range of bit-rates for the Barbara image. We conclude that the group testing technique is superior to the zerotree coding technique. In fact, our group testing algorithms also compare favorably to standards such as JPEG 2000, a benchmark algorithm that uses the dyadic wavelet decomposition. On the Barbara image, GTWP, as well as all the lapped transform algorithms, all perform better than JPEG 2000.

Our class definitions give one effective method of coding wavelet packets and block transform coefficients. They give more insight into the statistical characteristics of these coefficients, insight which can be may be used to create better image coders in the future. Thus, our image coders based on group testing are an interesting addition to the field of image compression.

6. Uncited references

[9,14]

Acknowledgements

We thank H.S. Malvar for providing code to compute the DCT, LOT, and LBT transforms. We also thank T. Tran for providing code for computing the GLBT transform.

References

[1] S.O. Aase, T.A. Ramstad, On the optimality of nonunitary filter banks in subband coders, IEEE Trans. Image Process. 4 (12) (December 1995) 1585–1591.

[2] M. Adams, JasPer: A software based JPEG-2000 codec implementation, http://www.ece.ubc.ca/˜mdadams/jasper.

[3] J. Andrew, A simple and efficient hierarchical image coder, in: Proceedings of the 1997 International Conference on Image Processing, Vol. 3, 1997, pp. 658–661.

[4] M. Antonini, M. Barlaud, P. Mathieu, I. Daubechies, Image coding using wavelet transform, IEEE Trans. Image Process. 1 (April 1992) 205–220.

[5] C. Brislawn, FBI/LANL wavelet/scalar quantization (WSQ) fingerprint image compression specification, ftp://wwwc3.lanl.gov/pub/misc/WSQ.

[6] R.R. Coifman, M.V. Wickerhauser, Entropy-based algorithms for best basis selection, IEEE Trans. Inform. Theory 38 (2) (March 1992) 713–718.

[7] G.M. Davis, S. Chawla, Image coding using optimized significance tree quantization, in: Designs, Codes, and Cryptography, 1997, pp. 387–396.

[8] E.S. Hong, R.E. Ladner, Group testing for image compression, IEEE Trans. Image Process. 11 (8) (August 2002) 901–911.

[9] E.S. Hong, R.E. Ladner, E.A. Riskin, Group testing for wavelet packet image compression, in: Proceedings of DCC 2001, Data Compression Conference, pp. 73–82.

[10] E.S. Hong, R.E. Ladner, E.A. Riskin, Group testing for block transform image compression, in: Thirty-Fourth Asilomar Conference on Signals, Systems and Computers, Vol. 1, 2001, pp. 769–772.

[11] H. Khalil, A. Jacquin, C. Podilchuk, Constrained wavelet packets for tree-structured video coding algorithms, in: Proceedings of DCC'99, Data Compression Conference, 1999, pp. 354–363.

[12] H.S. Malvar, Signal Processing with Lapped Transforms, Artech House, 1992.

[13] H.S. Malvar, Lapped biorthogonal transforms for transform coding with reduced blocking and ringing artifacts, in: 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 3, 1997, pp. 2421–2424.

[14] H. Malvar, Fast progressive coding without wavelets, in: Proceedings of DCC 2000, Data Compression Conference, March 2000, pp. 243–252.

[15] N.M. Rajpoot, F.G. Meyer, R.G. Wilson, R.R. Coifman, On zerotree quantization for embedded wavelet packet image coding, in: Proceedings of the 1999 International Conference on Image Processing, Vol. 2, 1999, pp. 283–287.

[16] K. Ramchandran, M. Vetterli, Best wavelet packet bases in a rate-distortion sense, IEEE Trans. Image Process. 2 (2) (April 1993) 160–175.

[17] J.M. Shapiro, Embedded image coding using zerotrees of wavelet coefficients, IEEE Trans. Signal Process. 41 (December 1993) 3445–3462.

[18] D. Taubman, High performance scalable image compression with EBCOT, IEEE Trans. Image Process. 9 (7) (July 2000) 1158–1170.

[19] T.D. Tran, R.L. de Queiroz, T.Q. Nguyen, Linear-phase perfect reconstruction filter bank: lattice structure, design, and application in image coding, IEEE Trans. Signal Process. 48 (1) (January 2000) 133–147.

[20] T.D. Tran, T.Q. Nguyen, A progressive transmission image coder using linear phase uniform filterbanks as block transforms, IEEE Trans. Image Process. 8 (11) (November 1999) 1493–1507.

[21] UCLA Image Communications Lab, images, http://www.icsl.ucla.edu/~ipl/psnr_images.html.

[22] G. Wallace, The JPEG still picture compression standard, Comm. ACM 34 (4) (April 1991) 30–44.

[23] M.V. Wickerhauser, Acoustic signal compression with wavelet packets, in: C.K. Chui (Ed.), Wavelets: A Tutorial in Theory and Applications, Academic Press, New York, 1992, pp. 679–700.

[24] Z. Xiong, O.G. Guleryuz, M.T. Orchard, A DCT-based embedded image coder, IEEE Signal Process. Lett. 3 (November 1996) 289–290.

[25] Z. Xiong, K. Ramchandran, M.T. Orchard, Wavelet packets-based image coding using joint space-frequency quantization, in: Proceedings of the 1994 International Conference on Image Processing, Vol. 3, 1994, pp. 324–328.

[26] Z. Xiong, K. Ramchandran, M.T. Orchard, Wavelet packet image coding using space-frequency quantization, IEEE Trans. Image Process. 7 (6) (June 1998) 892–898.