IMAGE COMPRESSION WITH MULTITREE TILINGS.

Y. Huang, I. Pollak, and C.A. Bouman

Purdue University
School of Electrical and Computer Engineering
West Lafayette, IN 47907

ABSTRACT

We use the framework of multitree dictionaries introduced in [9], to design a novel DCT-based image coder which significantly outperforms both the standard JPEG and the quadtree-based approach of [13].

1. INTRODUCTION.

A number of research efforts have recently concentrated on developing adaptive algorithms for representing and approximating signals in overcomplete dictionaries. This paper addresses the *best basis problem*—or, more generally, the *best representation problem*: given a signal, a dictionary of representations, and an additive cost function, the aim is to select the representation from the dictionary which minimizes the cost for the given signal. This paradigm has been successfully used for problems in compression [11,13], estimation [4,5,10,12], and time-frequency (or space-frequency) analysis [6–9, 14].

The original papers on best basis search [2, 3] considered the wavelet packet bases and bases of local cosines on dyadic intervals. In each of these two cases, all the bases in the dictionary can be organized using a single tree: a dyadic tree in 1-D and a quadtree in 2-D. This organization was exploited in [2,3] to devise a fast recursive tree pruning algorithm to find the best basis for any additive cost function.

Since then, a number of efforts have sought to lift the restrictions that a fixed dyadic/quadtree structure imposes on the underlying dictionary. Search methods for various dictionaries that correspond to different sets of possible time-frequency or space-frequency tilings have been proposed, such as the double-tree algorithm [6], time-frequency trees [14, 15], space-frequency trees [7], adaptive Haar-Walsh tilings [11], anisotropic wavelet packets [1, 5], anisotropic cosine packets [1], and mixed isotropic/anisotropic packets [1].

In the present paper, we build on our results reported in [9] where we developed a new framework of multitree dictionaries. We use this framework to design a novel DCT-based image coder which significantly outperforms both the standard JPEG and the quadtree-based approach of [13]. The basic computational engine for this coder is our optimal rectangular tiling algorithm [9]. We start our discussion in Section 2 with a review of this algorithm. It is then extended in Section 3, to result in an efficient JPEG-like image compression algorithm which we experimentally compare to JPEG in Section 4.

This work was supported in part by a National Science Foundation CAREER award CCR-0093105 and a Purdue Research Foundation grant.

2. A FAST RECURSIVE TILING ALGORITHM.

We consider all images supported on a discrete rectangular domain $Q \subset \mathbb{Z}^2$. Suppose we are given an image f and would like to segment it into rectangular tiles $P_1, P_2, \ldots, P_d$ so as to minimize a cost which is equal to the sum of the costs of the individual tiles: $\sum_{i=1}^{d} c(P_i)$, where c is a cost function which is application specific and which depends on the image f.

We restrict our choice of tilings to those which can be obtained by recursively splitting rectangles into pairs of subrectangles. Such a splitting process can be represented as a binary tree whose root corresponds to the entire image and whose every node corresponds to a unique rectangular region of the image. We assign the cost given above to every tree t whose leaf nodes form the tiling $\{P_1, \ldots, P_d\}$:

$$\text{COST1}(t) = \sum_{P \in \text{leaves}(t)} c(P). \qquad (1)$$

We then search over all trees to find one of the trees with the smallest cost. The optimal tiling is then the leaves of this tree. Since our search space consists of multiple trees, we call it a *multitree dictionary*.

We now describe our efficient search algorithm. Let C_P^* be the cost of the optimal tiling for a rectangle P. In particular, the optimal cost for the entire image is $C_Q^* = \min_t \text{COST1}(t)$. Our search algorithm makes the following recursive call, starting with $P = Q$:

$$C_P^* = \min\{c(P),\ \min(C_{P'}^* + C_{P''}^*)\}, \qquad (2)$$

where the inner minimization is done over all ordered pairs of rectangles (P', P'') which partition the rectangle P.

The pseudocode for the search algorithm is shown in Fig. 1. The optimal tiling of P is denoted by $\mathcal{B}_P^*$. Fig. 1(a) shows the pseudocode for the recursive calculation of the optimal splits and corresponding costs which are stored in a global data structure TABLE. Once this piece of pseudocode is executed, the optimal tiling is constructed using the routine in Fig. 1(b) which is assumed to have access to the same global data structure TABLE.

3. IMAGE COMPRESSION.

3.1. Refinement: State Variables.

In our image coding algorithm, we allow the choice of several quantizers for encoding each tile. To model this choice, we introduce the concept of a *state variable*. To every tile P, we associate

```
(C*_P, s*_P) ← best_split(P) {
    if C*_P has been computed
        get C*_P and s*_P from the global data structure TABLE;
    else {
        s*_P ← ∅;              //Initialize best left child s*_P
        C*_P ← c(P);           //Initialize best cost C*_P
        for (P', P'') = a partition of P into two rectangles {
            (C*_P', s*_P') ← best_split(P');
            (C*_P'', s*_P'') ← best_split(P'');
            if C*_P' + C*_P'' < C*_P {
                s*_P ← P';                 //Update s*_P
                C*_P ← C*_P' + C*_P'';     //Update C*_P
            }
        }
        record C*_P and s*_P in the global data structure TABLE;
    }
    return C*_P and s*_P;
}
```

(a) Recursive calculation of the optimal splits and corresponding costs.

```
B*_P ← best_tiling(P) {
    get s*_P from the global data structure TABLE;
    if s*_P is the empty set
        B*_P ← {P};
    else
        B*_P ← best_tiling(s*_P) ∪ best_tiling(P\s*_P);
    return B*_P;
}
```

(b) Recursive generation of the best tiling.

Fig. 1. Pseudocode specification of a fast recursive search for the best rectangular tiling: (a) the recursive calculation of the optimal splits and the corresponding costs; (b) the recursive generation of the best tiling. It is assumed that both routines have access to the same global data structure TABLE. The optimal tiling $\mathcal{B}^*_Q$ of an image domain Q is obtained with $(C^*_Q, s^*_Q) \leftarrow$ best_split(Q), followed by $\mathcal{B}^*_Q \leftarrow$ best_tiling(Q).

a state variable x_P taking values in some finite set which, without loss of generality, we assume to be $\{1, 2, \ldots, X\}$ where X is some fixed integer. Each term of the cost function is now allowed to depend on the corresponding state variable—in other words, we replace the cost given in Eq. (1) with the following:

$$\text{COST2}(t) = \sum_{P \in \text{leaves}(t)} c(P, x_P). \tag{3}$$

We use the following recursive call to search for the best tree, which now includes searching for optimal states:

$$C^*_P = \min\{\min_{x_P} c(P, x_P), \ \min_{P', P''} (C^*_{P'} + C^*_{P''})\}, \tag{4}$$

where, in addition to searching over all partitions (P', P'') of P, we now also search over all possible values of the state variable x_P. The additional minimization over the state variables is implemented as a straightforward modification of the pseudocode in Fig. 1(a). The global minimum of $\text{COST2}(t)$ is then C^*_Q.

3.2. Multitree-JPEG Image Coder.

To obtain our multitree-JPEG image coder, we fuse our rectangular tiling algorithm with several aspects of the compression strategy in [13]. The input image is partitioned into square blocks; for each block, the optimal tiling is found, and every tile is encoded. Following [13], we assume that one of several quantizers can be used for each tile, and optimize our choice of the quantizer for each tile concurrently with the search for the optimal tiling, via the algorithm of Section 3.1. When looking for the best tiling and the best quantizers, we optimize with respect to the rate-distortion cost [13] $D+\lambda R$, where R is the number of bits it takes to encode the image, D is the total distortion, and λ is a parameter. In order to use the tiling algorithm of Section 3.1, we assume that the cost $D+\lambda R$ has the form (3): $D + \lambda R = \sum_P (D(P, x_P) + \lambda R(P, x_P))$, where x_P is the quantizer used for the tile P, $D(P, x_P)$ and $R(P, x_P)$ are the corresponding distortion and rate, respectively, and the summation is performed over all the tiles in the tiling. Since the cost is additive, our rectangular tiling algorithm of Fig. 1, with the modification discussed in Section 3.1, can be used to find the optimal tree and the best quantizer for each tile. For each tile, we follow a JPEG-like procedure which finds the DCT coefficients, quantizes, and entropy-codes the coefficients.

The optimal tree for each block is encoded as follows:

- one bit per node is used to indicate whether the node is an internal node or a leaf;
- $\lceil \log_2 X \rceil$ bits per leaf node are used to encode the state x;
- $\lceil \log_2 \text{SPLITS}_P \rceil$ bits are used to encode the split location for every internal node P, where SPLITS_P is the total number of possible split locations for the node P.

Note that the iterative procedure described in [13] can be used to adjust λ so as to minimize D subject to a fixed bit budget, and a similar procedure can be used to minimize the rate subject to a fixed distortion.

4. COMPRESSION EXPERIMENTS.

We compare our multitree-JPEG compression algorithm with a standard JPEG and with the quadtree-based algorithm of [13].[1] We test the algorithms on two images: a 512×512 image "barbara" and a 256×256 image "lenna". The corresponding sets of rate-distortion curves are shown in Fig. 2. In each figure, the rate in bits per pixel is plotted against the peak signal-to-noise ratio (PSNR). For each quadtree and multitree experiment, a target distortion was fixed, and the rate was minimized. Note that our multitree algorithm (dashdot) outperforms the standard JPEG (dash) by about 2-3 dB and the quadtree algorithm (solid) by about 0.5-0.7 dB at a fixed bit rate. Equivalently, the multitree algorithm represents compression savings of about 25-30% over the standard JPEG and about 10% over the quadtree algorithm, for a fixed PSNR.

In these experiments, we take the block size to be 16×16, and allow any partitions at multiples of 4 (i.e., the smallest possible cell size is 4×4). This means that, for each 16×16 block, we search over 68480 distinct tilings—this is in contrast to the quadtree method which only allows 17 distinct tilings, and the standard JPEG which only considers one tiling. While the number of possible tilings for our method is drastically larger, the number of distinct subrectangles of each block—which is what determines

[1] The rate-distortion curves we obtain for the JPEG and quadtree algorithms are different from those given in [13] since we use a somewhat different implementation of JPEG—for example, we use a different set of quantizers. However, the relative improvement of the quadtree algorithm over JPEG that we observe is similar to what is reported in [13].

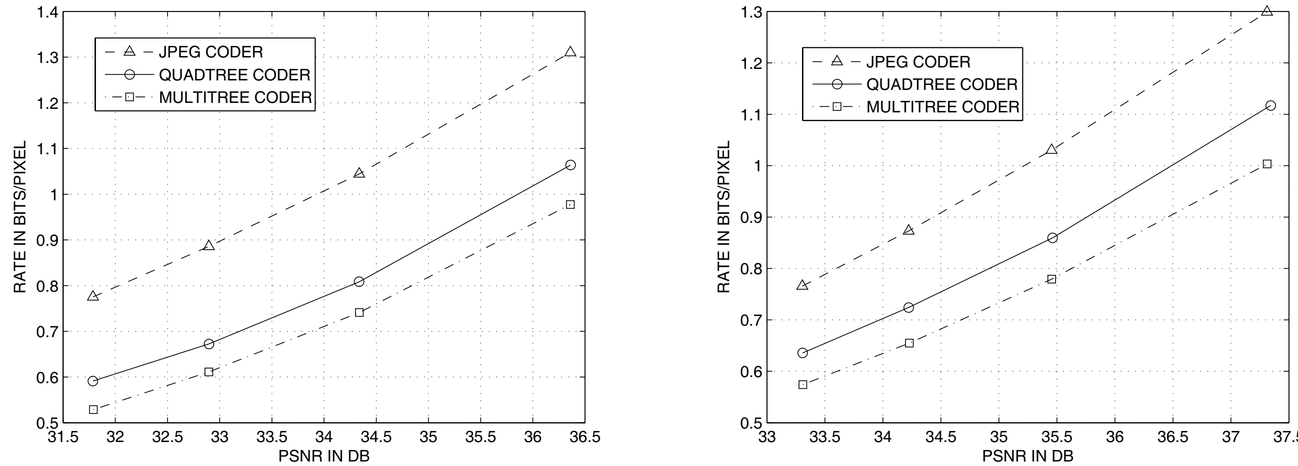


Fig. 2. Rate-Distortion curves for "barbara" (left) and "lenna" (right).

(a) Original image

(b) JPEG, 1.04 bpp

(c) Quadtree, 0.81 bpp

(d) Multitree, 0.74 bpp

Fig. 3. Results for the "barbara" image at PSNR = 34.3dB: (a) original image, (b) JPEG (rate = 1.04 bits per pixel), (c) quadtree compression (rate = 0.81 bits per pixel), and (d) multitree compression (rate = 0.74 bits per pixel).

the computational complexity of our algorithm—is only 100, compared to 21 for the quadtree method and 4 for the standard JPEG. Thus, we are able to search over a much larger set with only a modest increase in the computational burden.

The results for the "barbara" image at PSNR = 34.3dB are given in Fig. 3: JPEG, quadtree, and multitree compression algorithms achieve 1.04, 0.81, and 0.74 bits per pixel, respectively. Note that the images look basically the same; however, the multitree algorithm gives compression savings of 29% over JPEG and 9% over the quadtree algorithm.

Fig. 5 illustrates the results for the same image at the bit rate 0.49 bits per pixel. (In this experiment, the bit rate was fixed at 0.49, and the distortions for the quadtree and multitree methods were minimized.) At this bit rate, JPEG, quadtree, and multitree algorithm achieve PSNR for the overall image of 28.3 dB, 29.5 dB, and 30.4 dB, respectively. A patch from the image and its three compressed versions is shown in Figs. 4 and 5. In addition to a higher signal-to-noise ratio, it is clear from the figure that the multitree algorithm results in both less blocky renditions of homogeneous areas of the image, sharper edges, and less ringing and blockiness in the textured areas and around the edges.

5. CONCLUSIONS.

We applied our framework of multitree dictionaries and the accompanying efficient search algorithm [9], to image compression, and designed a new effective DCT-based image coder whose performance was illustrated through several examples. In the future, we plan to apply our framework to wavelet-based coders.

6. REFERENCES

[1] N.N. Bennett. Fast algorithm for best anisotropic Walsh bases and relatives. *J. of Appl. and Comput. Harmonic Analysis*, 8:86-103, 2000.

[2] R.R. Coifman, Y. Meyer, and M.V. Wickerhauser. Wavelet analysis and signal processing. In *Wavelets and Their Applications*, M.B. Ruskai *et al.*, Eds., pp. 153-178. Jones and Bartlett, Boston, 1992.

[3] R.R. Coifman and M.V. Wickerhauser. Entropy based algorithms for best basis selection. *IEEE Trans. Inf. Th.*, 38(2):713-718, March 1992.

[4] D.L. Donoho and I.M. Johnstone. Ideal denoising in an orthonormal basis chosen from a library of bases. *Comptes Rendus Acad. Sci.*, Ser. I 319:1317-1322, 1994.

[5] D. Donoho. CART and best-ortho-basis: A connection. *Ann. Stat.*, 25:1870-1911, 1997.

[6] C. Herley, J. Kovačević, K. Ramchandran, and M. Vetterli. Tilings of the time-frequency plane: construction of arbitrary orthogonal bases and fast tiling algorithms. *IEEE Trans. Sig. Proc.*, 41(12):3341-3359, Dec. 1993.

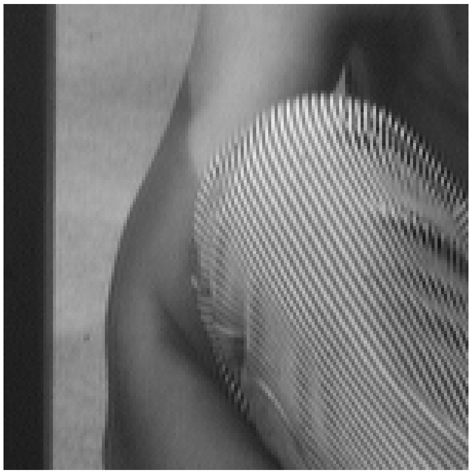


Fig. 4. A patch of the "barbara" image.

[7] C. Herley, Z. Xiong, K. Ramchandran, and M.T. Orchard. Joint space-frequency segmentation using balanced wavelet packet tree for least-cost image representation. *IEEE Trans. Im. Proc.*, 6(9):1213-1230, Sep. 1997.

[8] Y. Huang, I. Pollak, C.A. Bouman, and M.N. Do. New Algorithms for Best Local Cosine Basis Search. In *Proceedings* of ICASSP, May 17-21, 2004, Montreal, Quebec. www.ece.purdue.edu/~ipollak/icassp04.pdf

[9] Y. Huang, I. Pollak, M.N. Do, and C.A. Bouman. Optimal Tilings and Best Basis Search in Large Dictionaries. In *Proc. 37th Asilomar Conference on Signals, Systems, and Computers*, Nov. 9-12, 2003, Pacific Grove, CA.

[10] H. Krim and J.-C. Pesquet. On the statistics of best bases criteria. In *Wavelets and Statistics*, Lecture Notes in Statistics, A. Antoniadis, Ed., pp. 193-207. Springer-Verlag, 1995.

[11] M. Lindberg and L.F. Villemoes. Image compression with adaptive Haar-Walsh tilings. In *Wavelet Applications in Signal and Image Processing VIII*, Proc. SPIE 4119, 2000.

[12] P. Moulin. Signal estimation using adapted tree-structured bases and the MDL principle. In *Proc. IEEE-SP Int. Symp. TFTS*, pp. 141-143, Paris, June 1996.

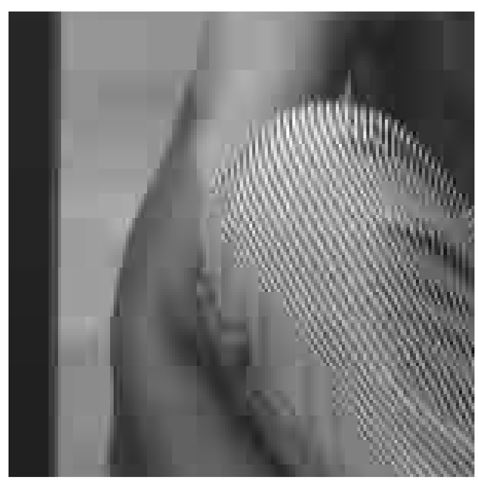
[13] K. Ramchandran and M. Vetterli. Best wavelet packet bases in a rate-distortion sense. *IEEE Trans. Im. Proc.*, 2(2):160-175, Apr. 1993.

[14] C.M. Thiele and L.F. Villemoes. A fast algorithm for adapted time-frequency tilings. *J. of Appl. and Comput. Harmonic Analysis*, 3:91-99, 1996.

[15] Z. Xiong, K. Ramchandran, C. Herley, and M.T. Orchard. Flexible tree-structured signal expansions using time-varying wavelet packets. *IEEE Trans. Sig. Proc.*, 45(2):333-345, Feb. 1997.

(a) JPEG, 28.3 dB

(b) Quadtree, 29.5 dB



(c) Multitree, 30.4 dB

Fig. 5. Results for the "barbara" patch at the bit rate of 0.49 bits per pixel: (a) JPEG (PSNR for the overall image = 28.3 dB), (b) quadtree compression (PSNR = 29.5 dB), and (c) multitree compression (PSNR = 30.4 dB).