COPERM: Transform-Domain Energy Compaction by Optimal Permutation

Nicholas D. Sidiropoulos, *Member, IEEE*, Marios S. Pattichis, Alan C. Bovik, *Fellow, IEEE*, and John W. Havlicek, *Student Member, IEEE*

***Abstract*—Compaction by optimal permutation (COPERM) is a tool for transform-domain energy compaction of broadband signals, whose foundation is a simple but powerful idea: Any signal can be transformed to resemble a more desirable (e.g., from a transform-domain compaction viewpoint) signal from a class of "target" signals (e.g., DCT basis functions) by means of a suitable permutation of its samples. One application of transform-domain energy compaction is in lossy compression. We pursue one possible thread in detail and demonstrate some interesting broadband image compression results.**

***Index Terms*—Broadband image compression, coding, energy compaction, permutation, textures.**

I. Introduction

THIS PAPER presents a new tool for transform-domain energy compaction of broadband signals based on a least squares permutation approach. Some basic theoretical results set the stage for the development of fast computational algorithms, which are exemplified in several experiments using a variety of broadband signals. These experiments demonstrate that the energy compaction afforded by COPERM is very significant—very few coefficients are usually sufficient to represent the compacted signal in the transform domain.

One application of transform-domain energy compaction is in lossy compression. The simplest way to compress a narrowband signal is to truncate its spectrum down to a few significant coefficients; the loss is small if the signal is sufficiently narrowband. In this context, the energy-compacting permutation carries most of the coding load as well as the coding cost. This approach will be shown to be fruitful for sufficiently broadband signals and images.

Manuscript received July 10, 1997; revised October 9, 1998. This work was supported in part by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under Grant F49620-93-1-0307 and by the Army Research Office under Contract DAAH 049510494. Part of this work was performed while the first author was with the Institute for Systems Research, University of Maryland, College Park. The associate editor coordinating the review of this paper and approving it for publication was Dr. Mahmood R. Azimi-Sadjadi.

N. D. Sidiropoulos is with the Department of Electrical Engineering, University of Virginia, Charlottesville, VA 22903 USA.

M. S. Pattichis was with the Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712 USA. He is now with the Department of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752 USA.

A. C. Bovik and J. W. Havlicek are with the Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712 USA.

Publisher Item Identifier S 1053-587X(99)03650-8.

Energy compaction is an important first step in almost all lossy compression techniques. Typical examples include the differencing step in adaptive differential pulse code modulation (ADPCM) or the prediction step in predictive PCM [1] and applying an energy-compacting transform, like DCT, as in JPEG [2], [3]. Prediction and/or transformation to some other more suitable domain are the two prevailing techniques for energy compaction. The former compacts most of the energy of the signal to be encoded in a relatively narrow range of amplitudes; the latter compacts most of the energy of the signal in a relatively narrow range of "frequencies" or, in general, transform coefficients. Both work well when the signal to be encoded is relatively smooth to begin with but are not very efficient in compacting the energy of persistent broadband signals. These signals occupy a wide bandwidth in the frequency domain and exhibit fast and persistent variation in the time domain. The purpose of this paper is to propose a new tool for energy compaction of broadband signals via optimal permutation. This permutation is matched to a given transform domain in the sense that its goal is to produce a permuted signal that resembles one of the associated basis functions as closely as possible in a least squares (LS) sense.

The idea of using permutations for (standalone) source coding has been investigated in the mid 1960's to early 1980's by Berger *et al.* [4]–[6]. This and other related work [7]–[10] is reviewed in the sequel (Section III). The idea of using a globally optimal permutation for effective transform-domain energy compaction of broadband signals is a contribution of this paper.

A. Organization

This paper is structured as follows. The main idea is introduced by example in Section II, which also contains some key results. Section III puts the main idea in perspective, connects with prior related work, and discusses some alternatives. Section IV describes an application of COPERM in broadband image compression, including several examples and a detailed discussion of results. Section VI summarizes current findings and presents some concluding remarks.

II. Main Idea

We introduce the main idea using the discrete Fourier transform (DFT). The DFT is closely related to the discrete Cosine transform (DCT), which is heavily used as a preprocessing energy-compaction block in many transform-domain codecs, like JPEG [2], [3]. These codecs also incorporate quantization and entropy encoding blocks. We temporarily leave these blocks out of consideration (although they too can be accounted for) and focus on energy compaction.

What is the best possible DFT input signal from an energy compaction viewpoint? Clearly, it is[1] *one* of the given DFT basis signals. Many real-life signals are smooth, thus having most of their energy in the lowpass band of the spectrum; the DFT is quite effective in compacting the energy of such signals. Actually, it is quite effective in compacting the energy of any narrowband signal, regardless of whether it is lowpass or bandpass.

The flip side of this is that the DFT is a poor choice for compacting the energy of broadband signals, e.g., textures, fingerprints, noise, fractals, or certain digital modulation signals. Given such a broadband signal, we would like to transform it such that

- the transformed signal is as narrowband as possible: ideally, one of the given DFT basis functions;
- the associated transformation is *invertible* so that we may recover the original signal from the transformed signal;
- the forward and inverse transforms are easy to *construct, apply,* and *represent.*

For the sake of simplicity, and without loss of generality, let us temporarily restrict our attention to so-called *constant modulus* signals. These are complex-valued signals whose magnitude remains constant over time, and only their angle changes. A constant modulus signal is any signal that can be written as $x(n) = Ae^{j\alpha(n)}$, where

A:	constant;
$\alpha(\cdot)$:	real-valued function;
j:	square root of -1.

Note that the *angle function* of $x(n)$ is not $\alpha(n)$ but rather $\alpha(n)$ modulo 2π. The reason for temporarily restricting ourselves to this class of signals is not technical but pedagogical: It allows us to make the first argument using the discrete Fourier transform, which is a very familiar tool.

Fig. 1(a) depicts the angle function of a broadband constant-modulus signal, Fig. 1(d) depicts the magnitude of the DFT of this signal, and Fig. 1(e) depicts the magnitude of the DFT of a suitable permutation of the given constant-modulus signal: Notice that the permuted signal essentially consists of a single DFT harmonic (there exist a few more negligible but nonzero DFT coefficients that are not visible in this plot). Fig. 1(b) depicts the angle function of the reconstructed signal obtained by setting all negligible DFT coefficients of the permuted signal to zero, computing the inverse DFT, and then depermuting using the inverse permutation. This effectively recreates the angle function of the original constant-modulus signal. Fig. 1(c) depicts the error between the angle functions of the original signal and the reconstructed signal [the signal-to-noise ratio (SNR) is about 100 dB].

Of course, from a compression standpoint, we have to consider the cost of storing not only the compacted transform (one complex number in this case) but the associated

[1]Modulo scaling by a (potentially complex) number, of course.

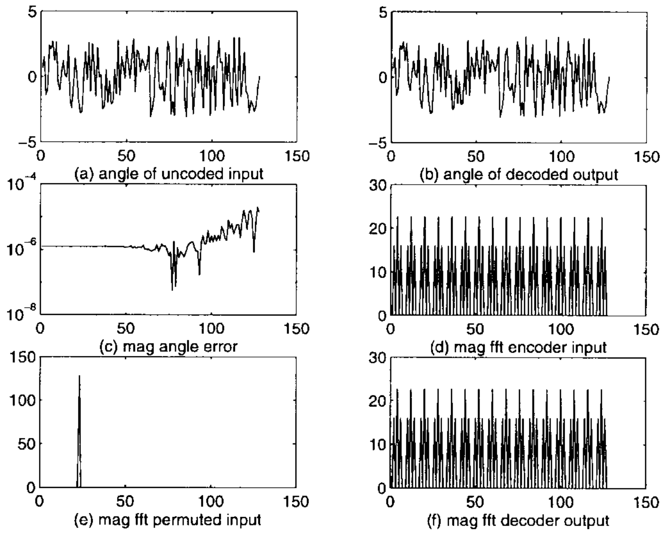


Fig. 1. $x(n) = e^{j(2\pi/N)(1+3n-2n^2+12n^3-n^6)}$.

permutation as well. There exist $N!$ possible permutations of a length-N signal—meaning that $\log(N!)$ bits are generally required to represent an arbitrary length-N signal permutation. This is slightly better than $N\log(N)$ [or $\log(N)$ bits per signal sample]. Thus, it appears that N should not be too big (actually, if the signal is digital, then $\log(N)$ should be strictly less than the number of bits used to represent a signal sample in order to achieve any compression gain at all). On the other hand, N should not be too small; otherwise, even the optimal matching is often not very accurate in absolute terms. It appears, then, that the choice of N exhibits an interesting (and unusual) tradeoff. However, as we will see later in Section IV, there is an elegant way to circumvent the restriction of having to work with relatively small N, even for digital data.

How do we find such a *suitable* permutation?

Recall that the sought transformation should ideally transform the given signal to a DFT basis function. In general, if we insist on perfect transformation to a basis function, then the transform cannot simultaneously be invertible as well as easy to construct, apply, and represent. In particular, if we restrict our attention to permutations, then we cannot insist on exact transformation because there exist signals that cannot be synthesized by permutation of a DFT basis function. It seems natural, then, to pose the following problem.

Given a constant modulus signal, find a permutation that best matches the angle function of the permuted signal to the angle function of *some* DFT basis signal, i.e., one that best matches the angle function of the permuted signal to $((2\pi/N)kn)$ mod (2π) for the best possible $k \in \{0, 1, \cdots, N-1\}$ in an LS sense.

Let us use the letters ϕ, θ, r to denote permutations of the integers in $\{0, 1, 2, \cdots, N-1\}$, i.e, $\{\phi(n), n \in \{0, 1, 2, \cdots, N-1\}\}$ is a permutation of $\{0, 1, 2, \cdots, N-1\}$ for some fixed N. Let $\mathcal{G}$ be the group of all such permutations ($|\mathcal{G}| = N!$).

In concise mathematical terms, we have the following.

Problem 1: Given the angle function $p(n)$ of a constant element modulus input signal, find

$$r^* = \mathrm{argmin}_{r \in \mathcal{G}} \min_{k \in \{0,1,\cdots,N-1\}} \sum_{n=0}^{N-1} \left| p(r(n)) - \left(\frac{2\pi}{N}kn\right) \bmod (2\pi) \right|^2. \quad (1)$$

Notice that in terms of the data sequence $p(n)$, this *is not* a projection problem—$p(r^*(n))$ need not be a permutation of some $((2\pi/N)kn) \bmod (2\pi)$ but rather as *close* to some $((2\pi/N)kn) \bmod (2\pi)$ as possible.

In solving this optimization, the following general theorems are key.

A. Key Theorems

Given a real-valued sequence $p(n)$ of length N, we will say that ϕ is a *sorting permutation* for $p(n)$ if the sequence $p(\phi(n))$ is sorted in increasing order (sorting permutations are not necessarily unique because of the possible existence of *ties*: Two or more elements may have exactly the same value).

Theorem 1 [4, Th. 1]: Given two *real-valued* sequences of length $N, p(n)$ and $\alpha(n)$, consider the following problem:

$$\begin{aligned} \mathbf{minimize:} \quad & \sum_{n=0}^{N-1} |p(r(n)) - \alpha(n)|^2 \\ \mathbf{subject\ to:} \quad & r \in \mathcal{G}. \end{aligned}$$

Let ϕ, θ be sorting permutations for $p(n), \alpha(n)$, respectively. Then, an optimum r is given by $r(\theta(n)) = \phi(n), \forall n$.

Proof: This result is [4, Th. 1] with a minor modification: In [4], one of the two sequences is already sorted. The outline is as follows. Let $q(n) = p(r(n)), \forall n$. An optimum r as postulated above registers the smallest value in $p(n)$ with the smallest value in $\alpha(n)$, and then, the second smallest value in $p(n)$ with the second smallest value in $\alpha(n)$, and so on. No other permutation can be better. To see this, consider any sequence $q(n)$ induced by a permutation that does not register the values as postulated above. Then, it is possible to find two indices, say n, m, such that $q(n) > q(m)$, but $\alpha(n) < \alpha(m)$. Then, a simple swap of $q(n), q(m)$ would reduce the squared error, and thus, the sequence induced by the permutation is not optimal. This can be easily seen by enumerating the (six) total orderings that are consistent with the constraints $q(n) > q(m)$, $\alpha(n) < \alpha(m)$. ■

Remark 1: Sorting is, at worst, an $O(N\log(N))$ operation [11]; hence, an optimum r can be found in $O(N\log(N))$ operations.

It is now clear how to solve the optimization problem in (1): For fixed k, invoke the above theorem to solve for the best r for the given k. This takes $O(N\log(N))$ operations. Repeat for all N different k's, and pick the best such r (for the best k) as the final answer. The overall process entails $O(N^2\log(N))$ operations.

Remark 2: Given that problems involving optimization over the permutation group are typically intractable [12], it is quite refreshing that our particular formulation (Problem 1) is not.

Later on, we will explain how we may reduce this figure to $O(N(\log(N))^2)$ without significant performance loss while at the same time significantly reducing the cost of storing the associated optimal permutation.

For finite-alphabet sequences, there exist more efficient sorting algorithms, like `binsort`, which is[2] $O(N)$ instead of $O(N\log(N))$ (the latter figure is also a worst-case lower bound for arbitrary real-valued data) [11]. As a result, the complexity of the overall process can be reduced to $O(N^2)$ for finite-alphabet sequences. Furthermore, there exists an interesting alternative to Theorem 1 for the special case of finite-alphabet sequences. This is explored next.

Theorem 2: Consider two *finite-alphabet* sequences of length N: $p(n)$ and $\alpha(n)$. Assume that the alphabets are fixed and known in advance [this can always be assumed for the "target" sequence $\alpha(n)$]. Let h_p and h_α be the histograms of the two sequences over the respective alphabets, and $M = \max(\mathit{length}(h_p), \mathit{length}(h_\alpha))$ (i.e., equal to the size of the largest alphabet). Then

$$\mathbf{minimum}_{r\in\mathcal{G}} \sum_{n=0}^{N-1} |p(r(n)) - \alpha(n)|^2 = f(h_p, h_\alpha) = f^*$$

and f^* can be computed from h_p and h_α in $O(M)$ operations.

Proof: From Theorem 1, the sought minimum is the squared Euclidean distance between the sorted sequences. Given a histogram, the computation of the associated sorted sequence is trivial; the same holds for the inverse operation. It remains to be shown that f^* can actually be computed from h_p and h_α in $O(M)$ [instead of a "naive" (for $M < N$) $O(N)$] operations. A simple algorithm would be as follows:

Algorithm 1:

```
make a copy of the histograms; initialize $f, l, m$ to 0;
repeat until all data are consumed:
{ increment each of $l, m$ until $h_p(l) > 0$ and $h_\alpha(m) > 0$;
  if $h_p(l) \geq h_\alpha(m)$ then
  { $f+ = h_\alpha(m) * |value_p(l) - value_\alpha(m)|^2$;
     $h_p(l)- = h_\alpha(m)$;
     $h_\alpha(m) = 0$;
  }
  else
  { $f+ = h_p(l) * |value_p(l) - value_\alpha(m)|^2$;
     $h_\alpha(m)- = h_p(l)$;
     $h_p(l) = 0$;
  }
}
```

where $value_p(\cdot)$ returns the value corresponding to its argument in the alphabet associated with the $p(n)$ sequence and similarly for $value_\alpha(\cdot)$; the resulting f equals f^*. ■

Remark 3: Of course, calculating the histogram of the input sequence is an $O(N)$ operation; thus, the fact that f^* can actually be computed from h_p and h_α in $O(M)$ [instead of $O(N)$] operations is of little utility when we are given $p(n)$ instead of h_p and is only interested in computing f^* for just one $\alpha(n)$. It is when we need to compare *several*

[2]Provided the alphabet is fixed and known in advance [11]. `binsort` is also known as `counting-sort`.

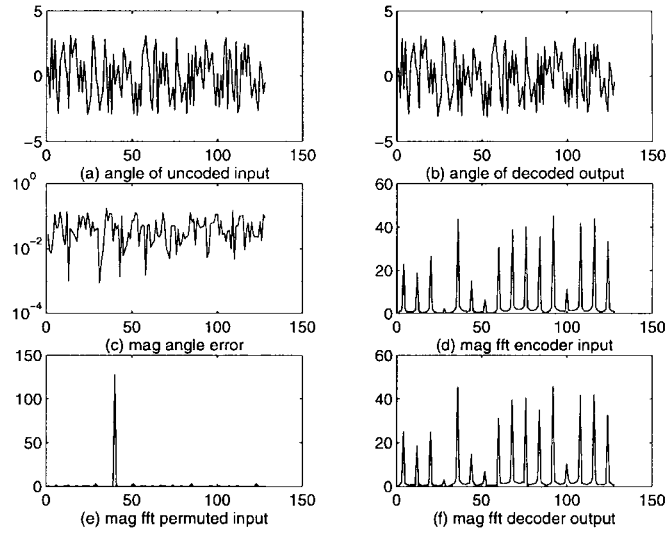


Fig. 2. $x(n) = e^{j(2\pi/N)(-n^{1/2}+1+3n-2n^2+12n^3)}$.

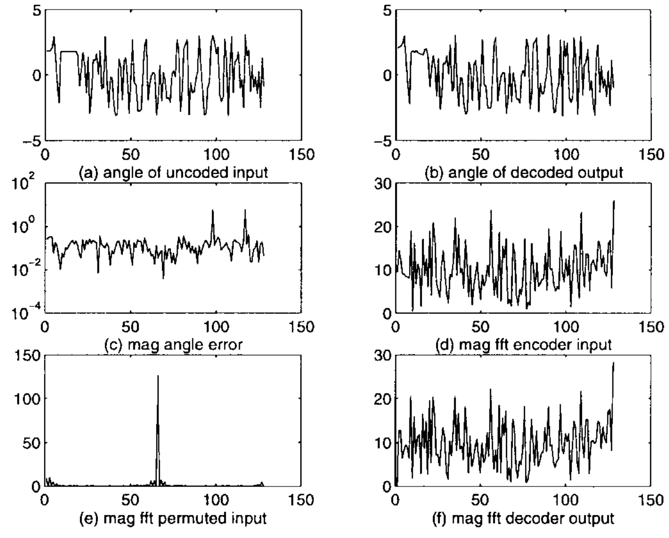


Fig. 3. $x(n) = e^{j(2\pi/N)(\Gamma(n)+36)}$. $\Gamma(\cdot)$ is the gamma function.

candidate target sequences $(\alpha(n))$ that the benefit afforded by this theorem becomes apparent.

In light of this latter theorem, an alternative approach emerges: Given a finite-alphabet input sequence and a number of target sequences, find the best target sequence by i) computing the histogram of the input sequence in $O(N)$ operations (the corresponding computation for the target sequences need only be done *once* and *off-line*) and ii) selecting the best target sequence by computing the associated f^* metrics, as above, at a complexity cost of $O(M)$ operations each. In the end, we only need *one* sorting to find the optimum match. This being a finite-alphabet sorting operation, it follows that the overall runtime complexity of this process is $O(N+T\times M+N) = O(N+T\times M)$ operations, where T is the number of target sequences, and M is maximum alphabet size. This would be beneficial for M small relative to N.

Notice that Theorem 1 provides the means to match two real-valued signals in the time domain by means of an optimal (in the LS sense) permutation. In the context of transform-domain coding, we would actually like to match the two signals in the specified transform domain. At first glance, the problem may appear tougher in the transform domain. For linear orthogonal transforms, however, LS matching-by-permutation in the time domain is equivalent to LS matching-by-permutation in the transform domain. Consider, for example, a real-valued sequence $x(n)$ of length N, where we let

$\boldsymbol{x}$: its vector representation;
D: $N\times N$ forward DFT matrix;
P: $N\times N$ permutation matrix;
$\boldsymbol{s}$: a desired spectrum.

Let $\|\cdot\|_2^2$ stand for squared Euclidean norm and ^{H} stand for Hermitian transpose. Then, $\|DP\boldsymbol{x}-\boldsymbol{s}\|_2^2 = \|P\boldsymbol{x}-(1/N)D^H\boldsymbol{s}\|_2^2$ by virtue of *Parseval's Theorem.*

B. Some Further Examples

Further examples are shown in Figs. 2(a)–(f) and 3(a)–(f), following the same format as Fig. 1. As in the first example, just *one* (the strongest) DFT coefficient is sufficient to adequately represent the optimally permuted signal.

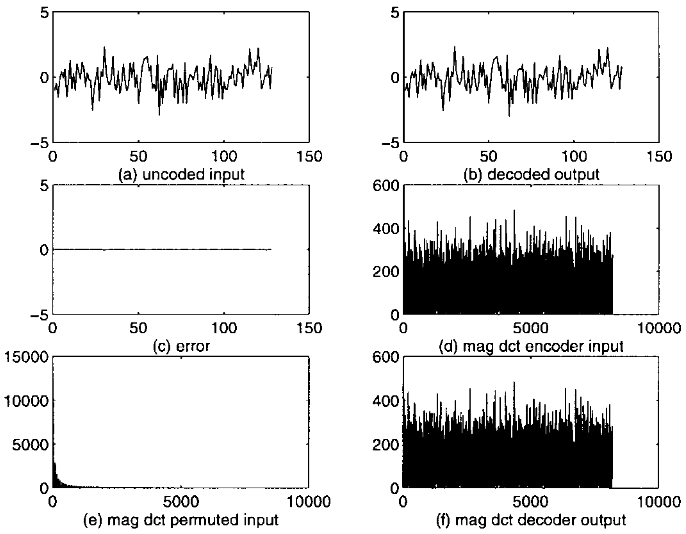


Fig. 4. $x(n)$ = a realization of a zero-mean, unit-variance white Gaussian process.

All examples so far involved the DFT and constant modulus signals, for which the optimal permutation was found by matching the angle function of the permuted signal to the angle function of the best possible DFT basis function in a LS sense. Of course, Theorem 1 is general, and, for real-valued signals, it can be applied to match the permuted signal itself to some other prescribed signal, or collection of signals, in an LS sense.

For real-valued signals, the DCT is often the transform of choice (e.g., [2] and [3]). Fig. 4(a)–(f) shows the result of applying Theorem 1 to a realization of a standard zero-mean, unit-variance white Gaussian process. The goal is to match the permuted signal to a DCT basis function in an LS sense. Fig. 4 follows the same presentation format as earlier figures. The differences here are as follows.

i) $N = 1024$.
ii) We use 100 DCT coefficients to approximate the compacted DCT of the permuted signal.

iii) Only the first 128 samples of the original, reconstructed, and error signals are shown to facilitate reader appreciation of the quality of reconstruction.

III. Perspective, Prior Work, and Alternatives in a Broader Context

Permutation codes are a class of codes that were introduced by Slepian [13] for reliably transmitting digital data over noisy communication channels. Permutation source codes were developed by Berger *et al.* [4]–[6], who made important contributions to the theory of permutation codes (see also Dunn [14]). The basic idea of permutation coding (as applied to the case of source coding) can be compactly stated as follows. We start with a basic "mother" codeword, which induces an associated codebook consisting of all distinct permutations of the mother codeword. Optimum encoding (and, in the case of channel coding, maximum likelihood decoding) is greatly simplified by Theorem 1. Given an input vector, the selected codeword is the permutation of the mother codeword that best matches the given input in a LS sense. This codeword is easily found by invoking Theorem 1. Relative to permutation source coding, COPERM has a different goal, namely, to create a narrowband signal in the DFT/DCT domain. Coupled with, e.g., simple truncation of the resulting narrowband spectrum, COPERM can be used to construct a compression algorithm. Such an algorithm may provide a more refined signal representation compared with permutation source coding, which represents the input using a single permutation of the mother codeword, which is the equivalent of using only the "carrier frequency."

A permutation of signal samples followed by the regular DFT is a discrete FM transform (DFMT) [15]–[17], and signal-dependent optimization of the permutation with the goal of obtaining a narrowband spectrum can be viewed as a signal-adaptive DFMT. The time-domain counterpart of this idea is to permute signal samples to minimize the sum of squared pairwise temporal differences. For real-valued signals, this amounts to a sorting permutation, and a variation of this idea has been explored by Neagoe [7], who proposed a predictive ordering/linear approximation technique for image data compression.

The idea of using a signal-adaptive block transform to enhance energy compaction and signal compression performance is not new, e.g., Caglar *et al.* [8] advocate using a bank of such transforms, generated by stacking signed permutations of signal centroid vectors. Relative to COPERM, the above approach employs a small number (e.g., 8) of signed permutations of a given centroid vector, and these are chosen to guarantee *orthogonality* of the resulting transform matrix instead of matching the input signal to a DFT/DCT basis function in a LS sense.

We may view a permutation of the elements of a raster-scanned vector of image samples as a *scan order*: an alternative (i.e., other than the given raster-scan) way of visiting the given collection of data points. The idea of using better scan orders to improve on prediction accuracy is quite old; the survey paper [9] provides a good recent summary of prior work in this direction. Work on alternative scan orders has focused on *local* scan orders and prediction-error estimates as a measure of goodness. Selection of the best scan order is accomplished either by exhaustive search or by minimum weight spanning tree (MST) algorithms. In contrast, our proposed technique focuses on global and globally optimum permutations, and the measure of goodness is Euclidean distance of the permuted signal from some prescribed (family of) target signal(s).

In a recent paper [10], Arnavut discusses an interesting approach to lossless image compression: treating the image as a *multiset permutation* and using a Lehmer inversion/image histogram representation of the image data. The Lehmer inversion vector typically has lower sample entropy than the image vector *per se*, and this facilitates lossless compression.

IV. Application in Broadband Signal/Image Compression

When using COPERM for DFT/DCT domain energy compaction in the context of lossy signal compression, the cost of coding the associated permutation and the cost of searching for the optimal permutation should be kept as low as possible. One way to contain the cost of coding the permutation is to exploit the inherent *periodicity* of DFT/DCT basis functions in the time domain.

Let us illustrate this point by focusing on the DFT basis function for $k = N/2$. This function is simply a binary oscillation: $(-1)^n$. When sorted, it gives two "buckets." Any sample of the original sequence may fall in *one* of the two buckets; exactly where it falls within a given bucket is irrelevant in terms of LS fit. Consequently, a single bit index per signal sample is sufficient to determine *an* optimal signal permutation for the given k. Stated another way, for $k = N/2$, there exist only 2^N distinct equivalence classes (with respect to LS fit) of permutations out of a total of $N!$ [essentially $2^{N\log(N)}$] permutations; thus, N bits [instead of $N\log(N)$ bits] are sufficient to represent the optimal permutation. We may decide on a class representative in an arbitrary fashion, provided it is consistent and known to both the encoder and the decoder. We simply sort indices that fall in the same bucket in increasing order. The decoder uses this convention to reconstruct the encoded permutation from its "bucketed" representation.

Similar (but smaller) savings can be realized for $k = N/4, \cdots, 2$. In general, we need $\log(N/k)$ bits per signal sample to represent an optimal permutation for $k = 1, 2, 4, \cdots, N/2$. We restrict our attention to these *dyadic frequencies* only. Of course, a similar argument can be made for the DCT, and we employ the 1-D DCT in the sequel.

According to Theorem 2, the best target frequency k is the one whose associated sample histogram is closest (in the LS sense) to the histogram of the signal at hand. Many broadband signals exhibit *level diversity*, meaning that smaller target frequencies k (which also exhibit higher sample level diversity) tend to provide better energy compaction. The flip side is that smaller k lead to higher $\log(N/k)$, i.e., more bits to code the permutation. One way to resolve this tradeoff is to pick the smallest k we can afford under a rate constraint

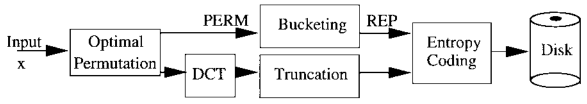


Fig. 5. COPERM-DCT encoder block diagram.

and then truncate the resulting narrowband spectrum down to a small fixed number of most significant coefficients. This approach is simple and fast (no frequency search), and it works very well for broadband signals that exhibit level diversity.

Fig. 5 presents the resulting COPERM-DCT coder in block diagram form. The input image is raster scanned to produce a long vector x of length N (N is assumed to be a power of 2). k is selected to be the smallest power of two for which $\log(N/k) \leq R$, which is a user-specified rate constraint. Typical values of k are $N/2, N/4, N/8$, resulting in 1–3 bits/sample of "raw" permutation coding overhead (this can be subsequently reduced by entropy coding). Let ϕ be a permutation that sorts x [ϕ is a byproduct of $sort(x)$; $x(\phi(n))$ is sorted in increasing order]. ϕ is partitioned into N/k length-k sub-blocks ("buckets"). Each of these sub-blocks is sorted in increasing order, in accordance with our earlier discussion and associated convention, and the result is stored in π. Let θ_k be a permutation that sorts the kth (1-D) DCT basis function (θ_k can be precomputed). Note that π is as good as ϕ in matching the permuted signal to the sorted samples of the kth DCT basis function. Applying θ_k^{-1} to π gives the total permutation p_t

$$p_t(\theta_k(n)) = \pi(n).$$

This is a permutation that best matches the input to the kth DCT basis function. A bucketed representation of π is then derived according to $r(\pi(n)) = floor(n/k), n = 0, 1, \cdots, N-1$. Note that given our intra-bucket sorting convention, r allows complete recovery of π (and therefore p_t), yet its samples take values in $0, \cdots, (N/k) - 1$, thus requiring $\log(N/k)$ b/sample to represent.

The input vector is permuted using p_t and transformed using DCT. A small user-specified number (e.g., 100) of most significant (strongest) DCT coefficients are selected and stored as (*frequency, coefficient*) pairs—the coefficient part being stored in 32-b floating-point representation. This information, along with the bucketed permutation representation r, plus header information (image size, bit depth, $\log(N/k)$, etc.) is stored in a file, which is subsequently processed by a lossless entropy coder (`gzip` in our particular implementation). Encoder complexity is $O(N\log(N))$. Recall that $N =$ image rows $\times$ image columns so that N can be big; however, the two basic operations involved are i) sorting and ii) DCT; both can be very efficiently implemented. We provide actual timing results in the next section.

The decoder inverts the entropy coding step (`gunzip` in our implementation), recovers π from r and p_t from π, θ_k, and the relation $p_t(\theta_k(n)) = \pi(n)$, reconstructs the truncated spectrum of the permuted signal from the list of (*frequency, coefficient*) pairs, performs an inverse DCT to recover an estimate of the permuted signal, and depermutes it using p_t^{-1}. Decoder complexity is also $O(N\log(N))$.

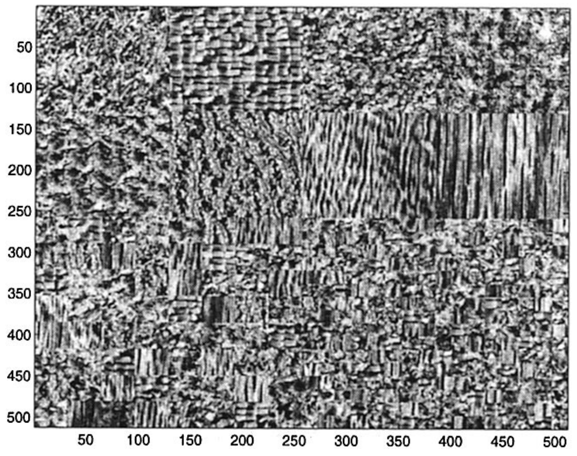


Fig. 6. Eight bit per pixel, 512 $\times$ 512 `texmos1.tiff` texture mosaic image number 1 from the USC-SIPI Image Database (http://sipi.usc.edu/services/database).

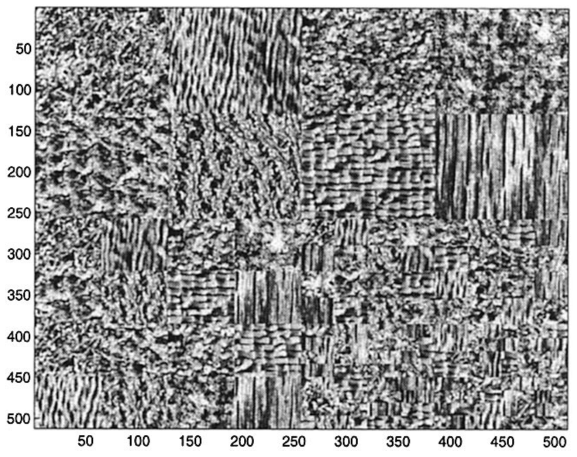


Fig. 7. Eight bit per pixel, 512 $\times$ 512 `texmos2.p512.tiff` texture mosaic image number 2 from the USC-SIPI Image Database (http://sipi.usc.edu/services/database).

Our implementation of the COPERM-DCT encoder/decoder is in interpreted `MATLAB` on `UNIX`. This implementation, plus the test images, some supporting programs, and a `README` file, are available at http://www.people.virginia.edu/˜nds5j/Welcome.html.

V. Experiments in Texture Image Compression

Textures are good examples of broadband image data. Texture coding is important in segmentation-aided image compression [18]. Figs. 6 and 7 depict two texture mosaic images from the USC-SIPI Image Database (http://sipi.usc.edu/services/database, files `texmos1.tiff`, and `texmos2.p512.tiff`, respectively). These are 8 b/pixel 512 $\times$ 512 collages of different types of texture.

As an assay, we downloaded the IJG implementation of JPEG (`cjpeg/djpeg`, which is available at ftp://ftp.uu.net/graphics/jpeg/jpegsrc.v6a.tar.gz), and the latest (updated after the November 1997 meeting of ISO/IEC JTC1/SC29/WG1) lossless JPEG codec (`locoe/locod V.0.90`, which is available for download at http://www.hpl.hp.com/loco/locodown.htm).

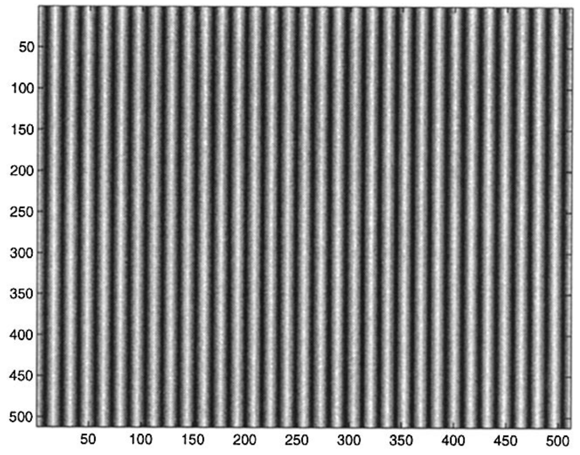


Fig. 8. Optimally permuted `texmos2.p512.tiff` texture mosaic image $N/k = 8$.

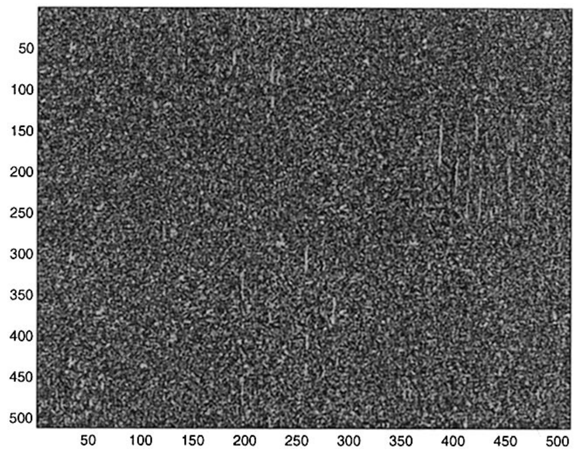


Fig. 10. Error image for COPERM-DCT encoded `texmos2.p512.tiff` texture mosaic image at a rate of 1.42 b/pixel.

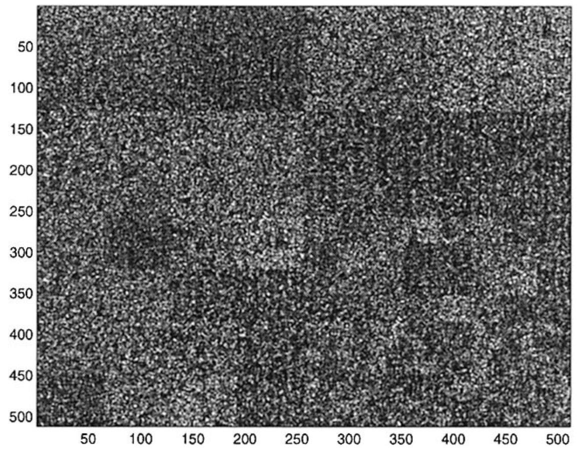


Fig. 9. Error image for JPEG encoded `texmos2.p512.tiff` texture mosaic image at a rate of 1.40 b/pixel.

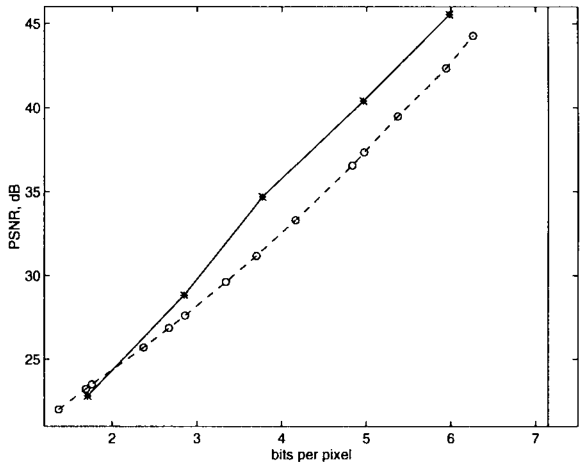


Fig. 11. Rate-distortion curves for `texmos1.tiff` texture mosaic image. Solid with "*" = COPERM-DCT; dashed with "o" = JPEG (`cjpeg -opt/djpeg`). The solid vertical line depicts the rate achieved by the latest lossless JPEG codec `locoe/locod V.0.90`. © Hewlett-Packard Co., 1997.

Consider the `texmos2.p512.tiff` image in Fig. 7. Fig. 8 depicts the optimal permutation of `texmos2.p512.tiff` for $N/k = 8$. It is readily verified that the permuted image has been effectively transformed into an almost-perfect harmonic.

Fig. 9 depicts the absolute error between the original `texmos2.p512.tiff` image and the JPEG (`cjpeg -opt/djpeg`) encoded/decoded reproduction at a rate of 1.40 b/pixel. Fig. 10 depicts the absolute error between the original `texmos2.p512.tiff` image and the COPERM-DCT encoded/decoded reproduction at a rate of 1.42 b/pixel (the same colormap is utilized for both renditions). Notice that the COPERM-DCT error image exhibits smaller overall error but also far less noticeable blocking.

Fig. 11 presents plots of rate-distortion results for the `texmos1.tiff` image. The horizontal axis is b/pixel, whereas the vertical axis is the peak signal-to-noise ratio (PSNR) measured in decibels:

$$\mathrm{PSNR} = 10\log_{10}\frac{255^2}{\mathrm{MSE}}$$

where 255 is the peak image amplitude, and MSE stands for mean squared error. Similarly, Fig. 12 presents plots of rate-distortion results for the `texmos2.p512.tiff` image. The same results are presented in tabular form in Tables I and II, respectively.

Some comments are in order. From Fig. 11 and Table I, which present results for the `texmos1.tiff` image, observe that at a rate of 2.85 b/pixel, the COPERM-DCT codec provides a PSNR margin of 1.2 dB's relative to JPEG with optimized Huffman tables. The gain increases with increasing bit rate and levels off at 3.5 dB's at roughly 3.75 b/pixel. Note that for this image, the latest lossless JPEG codec achieves a bit rate of 7.147 b/pixel, and PSNR results for lossy JPEG with optimized Huffman tables at about 1 b/pixel are only around 20 dB's; thus, the range of bit rates considered is meaningful if we are interested in high-quality texture rendition.

Further gains are possible. To see this, consider Fig. 12 and Table II, which present results for the `texmos2.p512.tiff` image. Observe that at a rate of approximately 1.40 b/pixel, the COPERM-DCT codec provides a PSNR margin of approximately 1 dB relative to JPEG with optimized Huffman tables. At 2.31 b/pixel, the

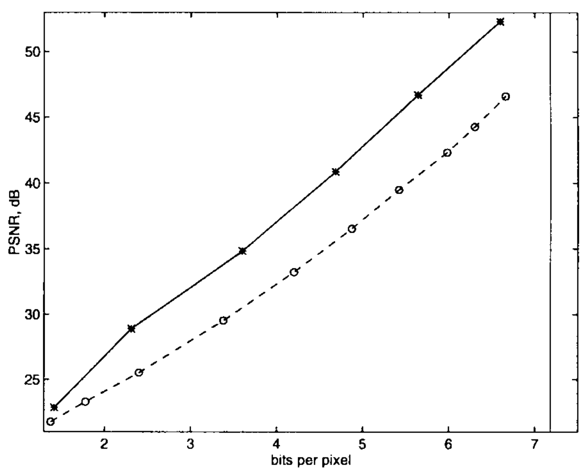


Fig. 12. Rate-distortion curves for `texmos2.p512.tiff` texture mosaic image. Solid with "*" = COPERM-DCT; dashed with "o" = JPEG (`cjpeg -opt/djpeg`). The solid vertical line depicts the rate achieved by the latest lossless JPEG codec `locoe/locod V.0.90`. © Hewlett-Packard Co., 1997.

TABLE I
RATE-DISTORTION PERFORMANCE ON THE 8 B/PIXEL, 512 × 512 `texmos1.tiff` TEXTURE MOSAIC IMAGE NUMBER 1 FROM THE USC-SIPI IMAGE DATABASE (http://sipi.usc.edu/services/database). PERMUTATION B/FREQUENCIES RETAINED FOR COPERM COLUMN (TOP TO BOTTOM): 2/100, 3/100, 4/100, 5/100, 6/100

COPERM	(using DCT)	JPEG	(cjpeg -opt/djpeg)	JPEG-LS	(locoe/locod)
bpp	PSNR	bpp	PSNR	bpp	PSNR
1.71	22.80	1.76	23.49	-	-
2.85	28.83	2.86	27.61	-	-
3.77	34.70	3.70	31.19	-	-
4.96	40.40	4.97	37.35	-	-
5.98	45.54	5.94	42.34	-	-
				7.147	∞

TABLE II
RATE-DISTORTION PERFORMANCE ON THE 8 B/PIXEL, 512 × 512 `texmos2.p512.tiff` TEXTURE MOSAIC IMAGE NUMBER 2 FROM THE USC-SIPI IMAGE DATABASE (http://sipi.usc.edu/services/database). PERMUTATION BITS/FREQUENCIES RETAINED FOR COPERM COLUMN (TOP TO BOTTOM): 2/100, 3/300, 4/150, 5/100, 6/100, 7/100

COPERM	(using DCT)	JPEG	(cjpeg -opt/djpeg)	JPEG-LS	(locoe/locod)
bpp	PSNR	bpp	PSNR	bpp	PSNR
1.42	22.83	1.38	21.76	-	-
2.31	28.87	2.40	25.53	-	-
3.60	34.84	3.38	29.50	-	-
4.68	40.85	4.20	33.24	-	-
5.64	46.69	5.42	39.48	-	-
6.60	52.31	6.66	46.59	-	-
				7.182	∞

gain is in excess of 3 dB's. Again, the gain increases with increasing bit rate. For this image, the latest lossless JPEG codec achieves a bit rate of 7.182 b/pixel, and PSNR results for lossy JPEG with optimized Huffman tables at about 1 b/pixel stand under 20 dB's.

In both cases, the gain in PSNR increases substantially with increasing bit rate. This is expected in view of our earlier discussion, and it suggests that even better PSNR gains for meaningful bit rates may be possible for high contrast-resolution (bit-deep) broadband images. From Tables I and II, it can be readily observed that i) only about 100 (out of 262 144) frequencies are sufficient to capture most of the energy in the permuted signal spectrum (thus, energy compaction works extremely well), and ii) the rate reduction benefit afforded by entropy coding the permutation representation is relatively small (about 25%), and it diminishes as we move toward higher bit rates. This extra rate margin *may* be sacrificed to further facilitate *transcoding* the encoded bitstream, as explained next.

A. Puncturing the Code and an Alternative Decoder

Each sample of the bucketed permutation representation specifies the bucket from which the corresponding reconstructed signal/image sample will be drawn. There is a total of N/k buckets associated with a given "center" dyadic frequency k. The samples of the inverse DCT of the truncated spectrum are sorted and split in N/k buckets ("percentiles"); the reconstructed signal/image samples are then drawn from the appropriate buckets. The samples associated with a given bucket are drawn in a *particular order*, namely, in increasing sample index order, in accordance with the convention adopted by the encoder.

Suppose that only the kth dyadic DCT coefficient is retained in the truncated spectrum. Each bucket of the corresponding basis function contains k identical samples. As a result, the particular order in which samples are drawn from each bucket is irrelevant. Now, consider the sorted and N/k-bucketed samples of the inverse DCT of a narrowband signal centered at frequency k. The corresponding buckets contain samples that are numerically close to each other. As a result, the particular order in which samples are drawn from each bucket is not significant. Thus, if energy compaction is successful, we need not reconstruct the exact permutation at the decoder. This observation may be exploited to build a simpler decoder. An additional advantage of such a decoder is that it may handle *scalable* transmissions. This is explained next.

In *multicast* systems, it is often desirable to fetch, e.g., 512 × 512 and 256 × 256 replicas of the same original image to two different user classes, respectively, each class having different quality of service constraints [19]. In *transcoding* [20], dropping the rate allocated to a particular transmission, due to changing bandwidth/traffic constraints, is often required. In this context, it is desirable to be able to reduce the aggregate data rate by simply dropping samples from the encoded data stream, rather than decoding the full 512 × 512 image and then re-encoding from scratch. Scalar quantization is particularly well suited for this task. The proposed COPERM-DCT approach is also well suited: Fig. 13 presents a 256 × 256 reproduction of the `texmos2.p512.tiff` image that was obtained by dropping samples from the encoded permutation representation stream produced for the full-resolution 512 × 512 image at the output of the COPERM-DCT encoder (b/pixel: 2.27, PSNR: 28.73 dB). Table III presents a rate-distortion performance comparison for 256 × 256 and 128 × 128 reproductions of the 512 × 512 `texmos2.p512.tiff` image that was obtained by dropping permutation representation samples from the COPERM-encoded bitstream (with entropy coding disabled) for the 512 × 512 image and then entropy coding the punctured bitstream. Observe that PSNR remains essentially unaffected, even for 4:1 downsampling in each dimension;

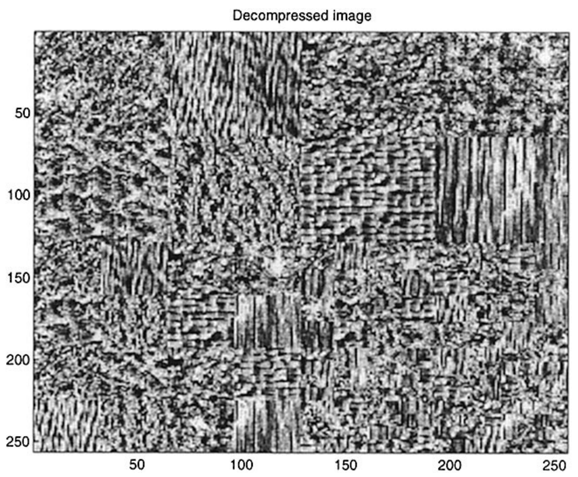


Fig. 13. Reproduction (256 × 256) of `texmos2.p512.tiff` texture mosaic image obtained by simply dropping samples from the encoded permutation representation stream produced for the full-resolution 512 × 512 image at the output of the COPERM-DCT encoder (b/pixel: 2.28, PSNR: 28.73 dB). The rate-distortion tradeoff remains essentially unaffected (cf., the first column of Table III).

TABLE III
PUNCTURING THE PERMUTATION REPRESENTATION: RATE-DISTORTION PERFORMANCE COMPARISON FOR TWO REPRODUCTIONS OF THE 512 × 512 `texmos2.p512.tiff` IMAGE, OBTAINED BY SYSTEMATICALLY DROPPING PERMUTATION REPRESENTATION SAMPLES FROM THE COPERM-DCT ENCODED BITSTREAM FOR THE 512 × 512 IMAGE. THE NUMBERS IN PARENTHESES IN THE FIRST ROW INDICATE PERMUTATION BITS/FREQUENCIES RETAINED

	COPERM-DCT	(3/100)	COPERM-DCT	(2/100)
	bpp	PSNR	bpp	PSNR
512x512	2.27	28.73	1.42	22.83
256x256	2.28	28.73	1.59	22.75
128x128	2.79	28.59	2.08	22.59

the bit per pixel rate is only mildly affected for up to 2 : 1 downsampling in each dimension.

Notice that in order to fully preserve the ability to change the sampling resolution via simple decimation of the encoded stream, the final entropy coding step should be disabled; however, the benefit of entropy coding the permutation representation is relatively small, as seen from Tables I and II; thus, this is a small price to pay if we are interested in a simple mechanism for aggregate rate reduction that effectively respects the original b/pixel-PSNR tradeoff.

The code available at http://www.people.virginia.edu/˜nds5j/Welcome.html includes two decoders: a basic decoder that reconstructs the exact permutation and a multirate decoder that is capable of decoding punctured code streams. The multirate decoder degrades average performance (relative to the basic decoder), but the degradation is usually barely noticeable, and, as discussed below, the multirate decoder is faster than the basic decoder.

B. Timing

The code available at http://www.people.virginia.edu/˜nds5j/Welcome.html is a high-level (interpreted) MATLAB implementation. It does not take advantage of finite-alphabet `binsort`. The encoder program is called `coperme`. There are two decoder programs: `copermd` and `mrcopermd`. The former reconstructs the exact permutation. In order to do so, it employs a loop that is *linear* in the size of the dataset yet relatively time consuming in MATLAB, which is slow in handling `FOR` loops. The latter works directly with the bucketed representation, as discussed above, and it is recommended in practice for large datasets since it avoids the aforementioned `FOR` loop.

For 256 × 256 images, both `coperme and copermd/mrcopermd` consume about 10–15 s of CPU time measured on a `SUN ULTRA-1`. For 512 × 512 images, `coperme and mrcopermd` consume about 1 min of CPU time. We emphasize that the two basic operations involved are sorting and the DCT. With a DSP board and/or custom hardware (or a lower level implementation utilizing `binsort` on a workstation), much better benchmarks may be expected.

VI. CONCLUDING REMARKS

This paper has introduced COPERM, which is a tool for transform-domain energy compaction of broadband signals. COPERM may offer a rate-distortion performance gain when used as a precoder, followed by a transform-domain encoder, and applied to broadband signal compression. This application has been illustrated by using COPERM coupled with simple truncation of the resulting narrowband spectrum to compress texture images, and interesting improvements over JPEG have been reported. The results suggest that enhanced performance should be expected for bit-deep textures. Texture coding is important in segmentation-aided image compression [18]. On the other hand, the application of COPERM is obviously counterproductive for relatively smooth signals and images, like `Lena`, or even `Barbara`, which are already relatively narrowband down to the 8 × 8 block level.

Although the basic idea behind COPERM is independent of the particular transform domain chosen (and even signal dimensionality), the cost of encoding the optimal permutation can be prohibitively high if the associated basis functions do not exhibit the "bucketing" property. Periodic basis functions enjoy this property, but even though periodicity *per se* is not necessary, the issue of whether or not COPERM can be helpful when coupled with a given transform (often chosen to meet other design objectives) needs to be investigated on a case-by-case basis.

An appealing feature of COPERM is that it is relatively simple and balanced in terms of encoder–decoder complexity. Another interesting feature is the possibility to lower sampling resolution via simple decimation of the encoded stream, and this is useful in the context of multicasting/transcoding.

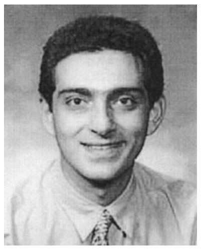
The connection with AM–FM signal analysis and synthesis is an interesting one; indeed, permutations spawn a class of discrete-time FM transforms that does not have a continuous-time counterpart. This class of discrete-time FM transforms is the subject of ongoing investigation.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for providing valuable feedback.

References

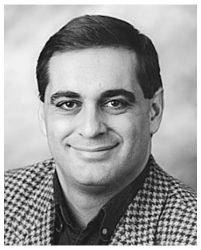
[1] B. Sklar, *Digital Communications*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
[2] G. K. Wallace, "The JPEG still picture compression standard," *Commun. ACM*, vol. 34, no. 4, pp. 31–44, Apr. 1991.
[3] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards—Algorithms and Architectures*. Boston, MA: Kluwer, 1996.
[4] T. Berger, F. Jelinek, and J. K. Wolf, "Permutation codes for sources," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 160–169, Jan. 1972.
[5] T. Berger, "Optimum quantizers and permutation codes," *IEEE Trans. Information Theory*, vol. IT-18, pp. 759–765, Nov. 1972.
[6] ——, "Minimum entropy quantizers and permutation codes," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 149–157, Mar. 1982.
[7] V.-E. Neagoe, "Predictive ordering and linear approximation for image data compression," *IEEE Trans. Commun.*, vol. 36, pp. 1179–1182, Oct. 1988.
[8] H. Caglar, S. Gunturk, B. Sankur, and E. Anarim, "VQ-adaptive block transform coding of images," *IEEE Trans. Image Processing*, vol. 7, pp. 110–115, Jan. 1998.
[9] N. D. Memon and K. Sayood, "Lossless image compression—A comparative study," *Proc. SPIE 2418 Still Image Compression*, pp. 8–20, 1995.
[10] Z. Arnavut, "Applications of inversions to lossless image compression," *Opt. Eng.*, vol. 36, no. 4, pp. 1028–1034, Apr. 1997.
[11] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*. Reading, MA: Addison-Wesley, 1983.
[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.
[13] D. Slepian, "Permutation modulation," *Proc. IEEE*, vol. 53, pp. 228–236, Mar. 1965.
[14] J. G. Dunn, "Coding for continuous sources and channels," Ph.D. dissertaion, Columbia Univ., New York, NY, 1965.
[15] M. S. Pattichis and A. C. Bovik, "AM-FM expansions for images," in *Proc. Euro. Signal Process. Conf.*, Trieste, Italy, Sept. 10–13 1996.
[16] J. P. Havlicek, "AM-FM image models," Ph.D. dissertation, Univ. Texas, Austin, 1996.
[17] M. S. Pattichis, "AM-FM transforms with applications," Ph.D. dissertation, Univ. Texas, Austin, 1998.
[18] O. J. Kwon and R. Chellappa, "Segmentation-based image compression," *Opt. Eng.*, vol. 32, pp. 1581–1587, July 1993.
[19] A. Banerjea, T. Waitian, and A. Zakhor, "A layered compression scheme for multicasting medical images across heterogeneous networks," *Proc. SPIE*, vol. 3031, pp. 265–276, 1997.
[20] S. J. Wee and B. Vasudev, "Splicing MPEG video streams in the compressed domain," in *IEEE Workshop Multimedia Signal Process.*, Princeton, NJ, June 1997. [Online] Available http://image.mit.edu/˜swee/hp.html.



Nicholas D. Sidiropoulos (M'92) received the Diploma in electrical engineering from the Aristotelian University of Thessaloniki, Thessaloniki, Greece, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland at College Park (UMCP), in 1988, 1990, and 1992, respectively.

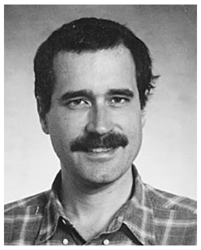
From 1988 to 1992, he was a Fulbright Fellow and a Research Assistant at the Institute for Systems Research (ISR), University of Maryland. From September 1992 to June 1994, he served his military service as a Lecturer in the Hellenic Air Force Academy. From October 1993 to June 1994, he also was a Member of the Technical Staff, Systems Integration Division, G-Systems Ltd., Athens, Greece. He has held Postdoctoral (1994 to 1995) and Research Scientist (1996 to 1997) positions at ISR-UMCP before joining the Department of Electrical Engineering, University of Virginia, Charlottesville, in July 1997. His research interests are in the broad area of signal and image processing.

Dr. Sidiropoulos received the NSF/CAREER award (Signal Processing Systems) in June 1998.



Marios S. Pattichis was born in Nicosia, Cyprus, on March 17, 1967. He received the B.S. (Hons.) degree in computer science and the B.A. (Hons.) degree in mathematics in 1991 and the M.Sc. and Ph.D. degrees in electrical engineering in 1993 and 1998, respectively, all from the University of Texas, Austin. For his undergraduate studies, he received a scholarship from the Cyprus-America Scholarship program.

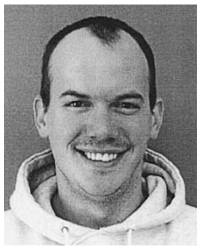
His research interests are in the broad area of signal and image processing, with an emphasis on AM–FM transforms and their applications, which was his dissertation topic. He is currently a Visiting Assistant Professor in the Department of Electrical Engineering and Computer Science, Washington State University, Pullman.



Alan C. Bovik (F'94) received the B.S., M.S., and Ph.D. degrees in electrical and computer engineering in 1980, 1982, and 1984, respectively, all from the University of Illinois, Urbana-Champaign.

He is currently the General Dynamics Endowed Fellow and Professor in the Department of Electrical and Computer Engineering, University of Texas, Austin, where he is the Associate Director of the Center for Vision and Image Sciences. During the Spring of 1992, he held a visiting position in the Division of Applied Sciences, Harvard University, Cambridge, MA. His current research interests include digital video, image processing, wavelets, and computational aspects of biological visual perception. He has published over 275 technical articles in these areas and holds U.S. patents for the image and video compression algorithms VPIC and VPISC.

Dr. Bovik is the recipient of the IEEE Signal Processing Society Meritorious Service Award (1998), the University of Texas Engineering Foundation Halliburton Award, and is a two-time Honorable Mention winner of the International Pattern Recognition Society Award for Outstanding Contribution (1988 and 1993). He has been involved in numerous IEEE professional society activities. Currently, he is on the Board of Governors of the IEEE Signal Processing Society, is the Editor-in-Chief of the IEEE TRANSACTIONS ON IMAGE PROCESSING, and is a member of the Editorial Board of PROCEEDINGS OF THE IEEE and several other journals. He was the Founding General Chairman of the *IEEE International Conference on Image Processing*, which was held in Austin, TX, in November 1994.



John W. Havlicek (S'97) received the B.S. degree in mathematics from The Ohio State University, Columbus, in 1987 and the Ph.D. degree in mathematics from Stanford University, Stanford, CA, in 1992. He is currently pursuing the Ph.D. degree in computer sciences at The University of Texas, Austin.

He was a Visiting Research Instructor with the Mathematics Department at Michigan State University, East Lansing, from 1992 to 1995 and a Visiting Assistant Professor with the Mathematics Department, Albion College, Albion, MI, from 1995 to 1996.