On the Representation of Wideband Images Using Permutations for Lossless Coding

M.S. Pattichis
The University of New Mexico
Dept of EECE and Center for High Performance Computing
EECE Bldg, Room 323B
Albuquerque, NM 87131-1356
pattichis@eece.unm.edu.

A.C. Bovik
The University of Texas at Austin
Dept of Electrical and Computer Engineering
Engineering Science Bldg, 540
Austin, TX 78712-1084
bovik@vision.ece.utexas.edu.

J.W. Havlicek
The University of Texas at Austin
Dept of Computer Sciences
Austin, TX 78712-1084
havlicek@cs.utexas.edu.

N.D. Sidiropoulos
University of Minnesota
Dept of Electrical and Computer Engineering
200 Union Street SE
Minneapolis, MN 55455
nikos@ece.umn.edu.

Abstract

We introduce a novel method for representing and coding wideband signals using permutations. The signal samples are first sorted, and then encoded using Differential Pulse Coding Modulation. We show that our method is optimal for DPCM coding and develop a novel algorithm for encoding the permutation information efficiently. We show that the new algorithm achieves coding gains over Huffman coding.

1 Introduction

The problem of representing wideband images is still very challenging. Spectral-based techniques like the Discrete Cosine Transform, and time-frequency techniques based on Discrete Wavelet Transforms are not very effective in coding such signals. For lossy coding, it is possible to rearrange the signal samples so as to approximate narrowband signals and hence enable the efficient use of fixed transform techniques like the DCT ([5], [4], [3]). In this approach, the number of possible permutations is reduced significantly by considering suitable "target", narrowband signals. Due to transform coefficient quantization, we cannot use the same approach for lossless coding. In general, it is not clear which permutation to chose for representing and encoding an arbitrary, wideband image.

For any signal, the problem of encoding arbitrarily large permutations appears to be intractable; in the worst case, growing as $\log_2(N!)$ with the number of samples. Hence, when we restrict our attention to encoding the permutation directly, the complexity growth forces us to restrict our attention to smaller signal segments.

It is shown that if the permuted signal is encoded using DPCM, the optimal permutation to use is one that sorts the samples in the signal.Here, we claim that the sorting permutation is optimal in the sense that the sample moments for the sorted samples are minimized. This leads to a natural method for encoding and decoding wideband images as described in Figure 1.

2 Problem formulation

Consider an arbitrary signal (or image)

$$X = \langle x_1, x_2, \ldots, x_N \rangle ,$$

where the signal samples x_i take on the possible values

$$v_1 \leq v_2 \leq \ldots \leq v_M$$

where each signal value v_i occurs with multiplicity n_i, for $i = 1, \ldots, M$. That is, in X, there are n_i occurrences of $x = v_i$.

For any signal, the problem of encoding arbitrarily large permutations appears to be intractable; in the worst case, growing as $\log_2(N!)$ with the number of samples. Hence, when we restrict our attention to encoding the permutation directly, the complexity growth forces us to restrict our attention to smaller signal segments. It is important to note that this limitation can be removed, and in fact the algorithms that we are considering are very general, provided that we allow ourselves to use the fact that the encoded sorted signal will share the same histogram as the original unsorted signal.

If we agree to first encode the sorted signal, followed by the permutation codes, then it is clear that the decoder could utilize knowledge of the distribution of the signal values. This scheme is extremely similar to transmitting a Huffman table, and then using the table to encode the signal. In the context of our presentation here, this is viewed as a particular case of minimum-entropy encoding for the permutation. Alternatively, we can view the problem as one that is closely related to scalar quantization. In scalar quantization, we select a range of values and a particular signal level for each level, and then quantize the input signal values to a particular level. To encode the sorting permutation, we would proceed in the same way that bucket sorting is used to sort a signal. The range of values will correspond to a "bucket" and the signal level will simply identify the "bucket" from all the others. Hence, as we see, the general framework that we are presenting here is very closely related to both Huffman coding and scalar quantization.

3 Sorting permutations are optimal for DPCM

We will now address two problems: (i) suggest a method for storing the sorted values and (ii) show that, among all possible permutation of the signal samples, the sorting permutation is optimal in that it is the permutation that minimizes the number of bits required for representing the signal.

We begin by posing the problem of coding the sorted signal. Given an arbitrary signal X, we sort its samples into X_P: $X_P = \langle x_{p(1)}, x_{p(2)}, \ldots, x_{p(N)} \rangle$, satisfying $x_{p(1)} \leq x_{p(2)} \leq \ldots \leq x_{p(N)}$, where:

$$\begin{pmatrix} p(1) & p(2) & \ldots & p(N) \\ 1 & 2 & \ldots & N \end{pmatrix} \tag{1}$$

denotes the permutation that takes the $p(i)$-th sample of the original signal into the i-th sample of the sorted signal. In what follows, we will refer to P^{-1} as the *sorting permutation*. We note that P^{-1} takes the i-th sample of the original signal into the $p^{-1}(i)$-th sample of the sorted signal. The sorted signal is simply:

$$\overbrace{v_1, \ldots, v_1}^{n_1 \text{ times}}, \overbrace{v_2, \ldots, v_2}^{n_2 \text{ times}}, \ldots, \overbrace{v_M, \ldots, v_M}^{n_M \text{ times}}$$

To encode the sorted signal, we want to apply DPCM between any two successive values. We recognize that there will be many repetitions in the signal, and after DPCM, the repeated values will produce a sequence of zeros that we wish to encode efficiently using run-length encoding. For the zeros, we use the notation $[0,\ n_i - 2]$ to imply that we are using run-length encoding on the zeros, storing the number of zero-occurrences following each zero entry ($n_i - 2$ zeros). If $n_i = 1$, then no zeros are stored. Thus, the DPCM with variable length encoding of the signal can be represented as:

$$\begin{gathered} v_1, \ldots, v_M - v_{M-1}, [0,\ n_{iM} - 2] \\ n_{i1}, n_{i2}, \ldots > 1 \end{gathered}$$

It is important to note that variable length encoding of the difference signal does not need to deal with negative numbers. Hence, from the start variable-length encoding of the sorted signal will require one less bit per signal sample than the bit per signal sample for the standard DPCM method. With this observation in mind, we can provide a bound on the number of bits required for storing the values by the variable-length encoding method:

Lemma 1 (bound on the number of required bits) *For the number of bits s required to store the DPCM signal described in (2), we have:*

$$\begin{aligned} s &= \lceil \log_2 (v_1 + 0.5) \rceil + \cdots \\ &\quad + \lceil \log_2 (v_M - v_{M-1} + 0.5) \rceil \\ &\quad + \sum_{n_i \geq 2} \lceil \log_2 (n_i - 1.5) \rceil \\ &\leq 2M - 4 + \lceil 2 \log_2 (v_M + M) \rceil \\ &\quad + \lceil 2 \log_2(N) \rceil \end{aligned} \tag{2}$$

Proof: The proof is given in [3].

Remark 1 *In variable-length encoding, signal differences are encoded using two fields: (i) the number of bits required to represent the magnitude (and sign) of the signal,*

and (ii) the actual value of the magnitude. In deriving this bound, we do not account for the overhead required to specify the number of bits needed to specify the difference between samples.

We note that as the number of samples N increases our bound grows as $\log_2(N)$, while the number of bits for the original signal grows linearly in N.

We next seek to justify our decision to sort the samples. First, we note that the number of bits required to capture the dynamic range for DPCM-encoding X_P is always less than the same number of bits for the original signal:

$$\max_{2\le i\le N} \left| x_{p(i)} - x_{p(i-1)} \right| \le \max_{2\le i\le N} |x_i - x_{i-1}| \qquad (3)$$

Here, it is easy to see that (3) is satisfied for any ordering of the signal samples $x_{q(1)}, x_{q(2)}, \ldots, x_{q(N)}$, where Q denotes an arbitrary permutation. Second, we prove a more powerful result that shows that sorting followed by DPCM is indeed the most efficient way to encode the signal (for the natural assumption stated in the lemma):

Lemma 2 (optimal property of sorting+DPCM) *Assume that the number of bits required to encode the difference between two signal values is a non-decreasing function (denoted by $\rho(\cdot)$) of the difference between the samples (for signal values x, y, z):*

$$x \le y \le z \quad \Rightarrow \quad \rho(x, z) \ge \rho(x, y) \quad \textit{and} \quad \rho(x, z) \ge \rho(y, z)$$

Then, let Q denote an arbitrary permutation of the signal values. The sorting permutation is optimal in the sense that:

$$\sum_{i=2}^{i=N} \rho\left(x_{p(i-1)}, x_{p(i)} \right) \le \sum_{i=2}^{i=N} \rho\left(x_{q(i-1)}, x_{q(i)} \right) \qquad (4)$$

Proof: The proof is given in [3].

As stated earlier, the differences in the sorted signal are always positive. Therefore, for the sorted sequence, we need not store the $N-1$ sign bits and this saves us the cost of an additional bit per sample. Naturally, we could also sort the signal in descending order (instead of ascending order), and these two sorted sequences are the only ones for which the sign bits need not be stored. This shows that the bound in (4) is conservative.

3.1 Order statistics for sorted sequences

In this Section, we will compute the variance of the DPCM of a uniformly distributed sequence of iid random variables, and demonstrate the substantial reduction in the variance for the DPCM of the sorted signal. We will show that the variance of the DPCM of the sorted signal only grows as $1/N^2$ while it grows as $1/N$ for the unsorted signal. We continue to show that the DPCM of the sorted signal will always have minimum absolute moments among all permutations of the signal samples.

We are interested in a sequence of iid random variables [1]:

$$X_1, X_2, \ldots, X_n \quad \text{are iid with pdf } p(.) \text{ and cdf } P(.)$$

The Uniformly Distributed Case

As an example, we consider the case where the X_i are uniformly distributed in the interval $(0, 1)$. Then, the $g_{s,\,s-1}(.)$ simplify to [1, page 103]:

$$g_{s,\,s-1}(y) = N(1-y)^{N-1}, \quad 0 \le y \le 1 \qquad (5)$$

For the unsorted sequence, the pdf of the DPCM signal reduces to the familiar triangle function spanning from -1 to $+1$ with a height of $+1$. Clearly, the sorted pdf given in (5) is far more compact.

For the unsorted pdf, we have zero-mean and variance

$$\begin{aligned} \sigma_X^2 &= 2\int_0^1 y^2(1-y)dy \\ &= 1/6 \end{aligned} \qquad (6)$$

For the sorted sequence, we have:

$$\mu_{X_p} = N\int_0^1 y(1-y)^{N-1}dy \qquad (7)$$

$$= \frac{1}{N+1} \qquad (8)$$

and the standard deviation:

$$\sigma_{X_P} = N\int_0^1 \left(y - \frac{1}{N+1} \right)^2 (1-y)^{N-1}dy \qquad (9)$$

$$= \frac{N}{(N+1)^2(N+2)} \qquad (10)$$

We compare (10) to (8). For the sorted sequence, the variance grows as $1/N^2$ while for the unsorted sequence it grows as $1/N$. This observation motivates our study of the absolute moments.

Lemma 3 (Minimization of the Absolute Moments) *We consider the absolute moments:*

$$\rho(x_r, x_s) = \frac{1}{N}|x_r - x_s|^m, \quad m \in Z^+ \qquad (11)$$

$C = \text{encodeI}\,(\,C,\,X\,)\quad\{$
$(\,C,\,X_P\,) = \text{sort\_and\_encode\_permutation}(C,\,X);$
$X_D = \text{compute\_dpcm\_signal}\,(\,X_P\,);$
$C = \text{compress\_dpcm\_signal}\,(\,C,\,X_D\,);$
$\}$

$(\,C,\,X\,) = \text{decodeI}\,(\,C\,)\quad\{$
$(\,C,\,P\,) = \text{permutation\_decoder}(C);$
$(\,C,\,X_D\,) = \text{decompress\_dpcm\_signal}(C);$
$X_P = \text{reconstruct\_from\_dpcm\_signal}\,(\,X_D\,);$
$P^{-1} = \text{invert\_permutation}\,(\,P\,);$
$X = \text{apply\_permutation}\left(\,X_P,\,P^{-1}\right);$
$\}$

Figure 1. The general encoder/decoder functions for lossless signal compression using a sorting permutation.

Let Q denote an arbitrary permutation of the signal samples, while P denotes the sorting permutation. Then the sorted signal exhibits the minimum absolute moments:

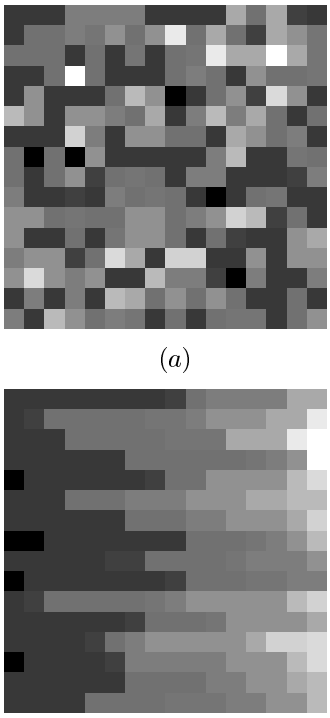
$$\frac{1}{N}\sum_{i=2}^{i=N}\left|x_{p(i-1)},\,x_{p(i)}\right|^m \leq \frac{1}{N}\sum_{i=2}^{i=N}\left|x_{q(i-1)},\,x_{q(i)}\right|^m,\; m \in Z^+ \tag{12}$$

Proof: The proof follows immediately from applying Lemma 2. It is easy to see that (11) satisfies (4). Thus, we substitute (11) into (4) to show that all the sample absolute moments of the sorted signal are less than the sample absolute moments of the unsorted signal: Q.E.D.

4 Algorithms for Encoding Optimal Permutations

In this subsection, we will present a simple rule for modifying most comparison based algorithms into a permutation coding algorithm. For a careful discussion of optimal sorting algorithms we turn to the classic work of [2]. In [2], the term *optimal algorithm* is reserved for algorithms that use the minimum number of comparisons for sorting a signal that has its values equally distributed. To encode a sorting permutation, we store the result of every comparison made in the sorting algorithm (using a single bit per comparison).

We apply this scheme for the three *asymptotically optimal* algorithms: *heap-sorting*, *merge-sorting* and *natural merge sorting*. In these algorithms, we encode the branches taken by each **if**-statement and use that information (only) to encode/decode permutations. Hence, these sorting algorithms lead to permutation encoding algorithms which are also asymptotically optimal.



(a)

(b)

Figure 2. In (a), we show a 16×16 image block of the sensor image. This is an 8-bit image. In (b), we show the same block, where each row of values was sorted in ascending order.

An example is shown in Figure $2(a)$. For each row, the samples are sorted in Figure $2(b)$. We consider two algorithms: (i) DPCM + Huffman coding, (ii) merge-sort permutation encoding and DPCM + Huffman coding of the sorted samples. For sorting every 4-samples, merge sorting requires 4.66 bits per pixel versus 5.11 bits per pixel for DPCM+Huffman, an 8.9% improvement. For sorting every 8-samples, merge sorting requires 4.80 bits per pixel. For sorting every 16-samples, merge sorting requires 4.85 bits per pixel. Hence, for this 8-bit image, sorting is best for applying on every four samples.

5 Conclusion

We introduced a novel representation for wideband images based on encoding permutations. We have established that our method is optimal for DPCM coding. Also, we developed and implemented optimal encoders/decoders for our permutation method, and demonstrate coding gains over

traditional DPCM+Huffman techniques. Since permutation encoding only depends on the number of samples, our method is very promising for wideband images with a significant number of bits per sample. Future research will focus on extending the method to multispectral images.

References

[1] H. A. David. *Order Statistics*. Wiley, New York, second edition, 1981.

[2] D.E. Knuth. *The Art of Computer Programming*, volume 3 Sorting and Searching. Addison-Wesley, Menlo Park, California, first edition, 1973.

[3] M. S. Pattichis. *AM-FM Transforms with Applications*. PhD thesis, The University of Texas at Austin, 1998.

[4] M.S. Pattichis, A.C. Bovik, J.W. Havlicek, and N.D. Sidiropoulos. Multidimensional orthogonal fm transforms. *IEEE Trans. on Image Processing*. in revision.

[5] N. D. Sidiropoulos, M. S. Pattichis, A. C. Bovik, and J. W. Havlicek. Coperm: Transform-domain energy compaction by optimal permutation. *IEEE Trans. on Image Processing*, 47:1679–1688, June 1999.