Stack-Run Image Coding

M.J. Tsai, J.D. Villasenor and F. Chen

mjtsai@icsl.ucla.edu, villa@icsl.ucla.edu, fchen@icsl.ucla.edu

This paper was published in the IEEE Transactions on Circuits and Systems

for Video Technology, vol. 6, pp. 519-521, Oct. 1996

Abstract

We describe a new image coding approach in which a 4-ary arithmetic coder is used to represent significant coefficient values and the lengths of zero runs between coefficients. This algorithm works by raster scanning within subbands, and therefore involves much lower addressing complexity than other algorithms such as zerotree coding that require the creation and maintenance of lists of dependencies across different decomposition levels. Despite its simplicity, and the fact that these dependencies are not explicitly utilized, the algorithm presented here is competitive with the best enhancements of zerotree coding. In addition, it performs comparably with adaptive subband splitting approaches that involve much higher implementation complexity. Finally, although this technique is described here in the context of a wavelet coding system, it can also be applied in a block DCT framework.

I. Introduction

Wavelets and other multiresolution techniques have received significant attention as a means to perform efficient coding of images and other multidimensional data. The principle behind the wavelet transform [1], is to hierarchically decompose an input signal into a series of successively smaller subbands. At each level in a transformed image, the low-pass "reference" subband and the three associated detail subbands contain the information needed to reconstruct the subband at the next higher resolution level. Decomposition can be performed either in a dyadic manner on the reference signals alone, or can be applied in a more general recursive manner on any of the high-pass or low-pass subbands. Efficient image coding is enabled by allocating bandwidth according to the relative importance of the information in the subbands, and then quantizing the transformed data values.

The advantages of wavelet-based coding over block-based discrete cosine transform algorithms such as JPEG have been known for several years, and more recent efforts have aimed to improve wavelet coding through innovations either on the transform or on the subsequent quantization. On the transform side, researchers have explored the issues of wavelet filter design and selection [1, 2] and have introduced techniques to perform subband splitting and basis selection adaptively as a function of local image characteristics [3, 4]. Quantization and entropy coding of wavelet transformed images has been performed using traditional scalar and vector quantization techniques, as well as through a class of zerotree approaches that utilizes the dependencies across subbands [5]. Refinements of zerotree that consider the rate-distortion tradeoffs in tree construction and more efficient representation of the zerotree data structures have further improved performance [6, 7]. While there are inevitably caveats associated with the use of mean square error metrics, the combination of a universal error metric with a de-facto standard set of test images has provided an important vehicle for presentation and evaluation of new coding results. Measured using peak signal to noise ratio (PSNR), the combination of adaptive subband splitting and zerotree coding has produced the best coding results reported in the literature to date [4]. For the work presented here we also report results in terms of PSNR, which we have found a reliable indication of visual quality for comparisons with the previous coding work in [4], [5], [7] and elsewhere.

The principal result of the present paper is a new data structure for encoding quantized transformed image data that has the simultaneous advantages of being 1) very simple conceptually and computationally, and 2) giving coding performance that is better than the basic zerotree algorithm and competitive with its most recent refinements. At the highest level, this algorithm, which we refer to as "stack-run coding", involves a wavelet transform, scalar quantization, run-length, and arithmetic coding, all of which are well-established techniques. The main innovation lies in the construction of a symbol set for the arithmetic coding of run/level pairs in which context information is used to enable multiple uses of a single symbol. When coupled with application of rate-distortion optimizations applied to the smallest reconstruction levels of the quantizer, this yields a low complexity coding scheme which consistently outperforms zerotree coding by over 1dB PSNR at .25 bits/pel (bpp) for typical grayscale images.

II. Stack-run coding

Consider a wavelet transformed image to which a uniform scalar quantizer is applied, with the same quantizer step size used across all subbands. The quantized wavelet coefficients can be partitioned into two groups containing zero-valued and non-zero valued (referred to as significant) coefficients respectively. Let each significant coefficient be represented in binary notation as a stack, or column of bits with the MSB at the top and the LSB at the bottom. The binary information is for now assumed to be unsigned, with sign information to be considered later. This binary representation is illustrated in Figure 1, which shows a subband with three significant coefficients having values 35, 4, and 11. For reasons that will be explained below, the binary representation is incremented by one with respect to the value; e.g. 4 is shown as binary 101 as opposed to binary 100.

A complete description of a subband can be provided by starting in one corner of the subband

and performing a raster scan described in terms of pairs of the form (a, b) where a is the number of zero-valued coefficients encountered before the next significant coefficient, and b is the magnitude and sign of the significant coefficient. This concept of generating (stack, run) pairs is not new, and in fact is similar to the procedure used to generate the input symbol set for Huffman coding in the JPEG standard. However, in contrast with the JPEG approach, we design a symbol set that takes advantage of the fact that all nonzero binary numbers begin with 1, and therefore that if the binary representation of a significant coefficient value is ordered from least significant bit (LSB) to most significant bit (MSB), the MSB does not need to be explicitly encoded if some other means of terminating the binary word can be found. Similarly, the run lengths also have a binary representation, the first bit of which contains no information since its value must always be 1. In addition, the run length values are always positive.

Because of the need to distinguish between (nonzero) level values and runs of zeros , a symbol set containing only "0" and "1" will not support efficient coding. Instead, we use an alphabet with the following four symbols and meanings: "0" is used to signify a binary bit value of 0 in encoding of significant coefficients. "1" is used for binary 1 in significant coefficients, but it is not used for the MSB. "+" is used to represent the MSB of a significant positive coefficient; in addition "+" is used for binary 1 in representing run lengths. "−" is used for the MSB of negative significant coefficients, and for binary 0 in representing run lengths.

During decoding, the use of the symbols in the alphabet removes any ambiguity between run lengths and coefficient values. In representing coefficient values, the symbols "+" and "−" are used simultaneously to encode both the sign of coefficients, and bit plane location of the MSB. An analogous gain in efficiency can be obtained in representation of the run lengths, which are ordered from LSB to MSB, and represented using the symbols "+" and "−". Since all binary run lengths start with 1, one can omit the final (e.g. MSB) "+" from most run-length representations without loss of information. The only potential problem occurs in representation of a run of length one which would not be representable if all MSB "+" symbols were eliminated. To circumvent this, it is necessary to retain the MSB "+" symbol for all runs of length $2^k - 1$, where k is an integer. The run length representations are terminated by the presence of a "0" or "1", which is the LSB of the next significant coefficient. No explicit "end-of-subband" symbol is needed since the decoder can track the locations of coefficients as the decoding progresses. Because the symbols "+" and "−" are used to code both runs and levels, a level of magnitude one, which would be represented only by the "+" or "−" that is its MSB, would be indistinguishable from a run. This is handled most easily by simply incrementing by one the absolute value of all levels prior to performing the symbol mapping.

The use of this alphabet is illustrated in Figure 2, which differs from Figure 1 only in that one of the significant coefficients is negative. If the raster scan is begun as indicated by the asterisk and arrow in the figure, 3 zero-valued coefficients will be encountered before the first significant coefficient. This string of 3 zeros corresponds to binary run length 11, which by one-to-one mapping with the symbols "+" and "−" would lead to the symbol string representation "++". As discussed before, since the length of the run satisfies $2^k - 1$ the final "+" (e.g. the MSB) is not dropped. The

next coefficient after the run of 3 zeros has decimal value 35. This is incremented by one to 36 and then ordered from LSB to MSB to give the representation "00100+". The next significant coefficient has value 4, and occurs immediately after the 35 (e.g. there is no run of zero-valued coefficients). This is represented by appending "10+" to the encoded symbol stream. This corresponds to the binary expression of decimal 5 in which the ordering is from LSB to MSB where the MSB has been replaced by "+". The run of length 10 that follows this coefficient could be represented using "$-+-+$", corresponding to binary 1010 ordered from LSB to MSB. In this case, however, the final "+" in this sequence is redundant, and this run length of 10 is represented as "$-+-$". The final coefficient has value -11 and is decremented to -12 and represented as "001$-$".

Given this method of mapping data into symbols, there are several of further steps that can be taken to enhance performance prior to arithmetic coding of the symbol set. First, the probability tables used in the arithmetic coder can be adaptive. This will take advantage of local variations in the run length and level statistics. Second, the symbols for the run lengths and the LSB of the levels can be considered separately from the remaining symbols for the levels, resulting in two probability tables that can be independently adapted. Both tables will contain all four symbols because of the need to represent terminating information, but the run length table will be dominated by "$-$" and "+" and the level value table will be dominated by "1" and "0". Further performance enhancement at the cost of some increase in complexity can be realized if the symbols for the levels are scanned in bit plane order.

An additional optimization can be performed by examining rate-distortion tradeoffs. An analysis of the bitstream produced by arithmetic encoding of the partitioned (by wavelet subband and by run versus coefficient value) symbol set shows that the outer, higher frequency subbands dominate the bandwidth. This is because these subbands are larger and contain more significant coefficients after quantization. The rate penalty for encoding the smallest of these coefficients is high, because they often interrupt what would be large runs of zeros that could be very efficiently coded. In addition, the reduction in distortion contributed by these coefficients is usually very small. To realize improved rate-distortion performance we include the flexibility to truncate small coefficients to zero, effectively giving a wider zero bin to the quantizer. It is also possible to envision a more rigorously designed, rate-distortion optimal nonuniform quantizer optimized for each subband, but this would add significant complexity to the system.

III. Experimental results

We have applied stack-run coding to a large set of images across a wide range of bit rates. A low complexity adaptive arithmetic coder based on [8] was used to encode the symbol stream created using the approach described in the previous section. Coding results for the 512x512 grayscale "Lena" and "Barbara" images are summarized in Table 1. The wavelet filter used for the first row of the table was the 10-tap/18-tap biorthogonal filter given in Table 2. For the second row the 9-tap/7-tap filter [9] was employed. The table also contains results for zerotree using the algorithm of Shapiro [5], for adaptive frequency-space splitting based on the work of Xiong et al. [4], and for the enhanced zerotree algorithm of Said and Pearlman [7]. The table also indicates the subband

splitting used for each transform. In most cases, a 6-level octave band decomposition was used. For the "Barbara" image in the first and last rows of the table, a uniform 64 band decomposition with the lowest resolution band subject to additional 3 level octave band decomposition was performed. The performance of stack-run coding is consistently 1 to 2 dB above the original zerotree algorithm of [5]. It is slightly inferior to the algorithms of [4] and [7] when coding is performed using the same DWT and filter. In general, we have found that stack-run coding is most advantageous at relatively low bit rates, where the percentage of zero-value coefficients, and therefore of zero runs of significant length, is relatively high.

One metric that is often used to quantify algorithmic complexity is running time on a particular machine. While running times are strongly implementation dependent, and are vulnerable to machine-specific costs of different data and memory structures, they are useful for approximate comparisons. For the Lena image at .25 bpp using a 6-level DWT based on the 9/7 filters, the algorithm of Said and Pearlman requires 2.08 seconds on an HP735/125 workstation compared with 2.36 seconds for stack-run coding when the same filter bank and decomposition are used [10]. These times are for a round trip encode/decode cycle and include I/O, and forward and inverse transformation, quantization, and lossless coding. As neither of the implementations have been optimized for speed, further improvement is certainly possible.

IV. Conclusions

A new image coding algorithm based on 4-ary arithmetic coding of significant coefficients and run lengths of a transformed, scalar-quantized image has been proposed. The symbol meanings are dependent on context, and are encoded using context-specific probability tables. Further savings is enabled because the MSBs of the binary run length and significant coefficient values are not explicitly encoded, but are instead implied by the structure of the symbol stream. Application of this coding scheme to commonly used test images demonstrates consistently high performance. In addition, the stack-run coding scheme described here is fast and simple to implement in hardware, and supports progressive transmission because more important subbands can be sent earlier.

Although we explored this coding scheme using a wavelet transform with relatively simple decompositions, it could also be combined with other techniques such as adaptive subband splitting and adaptive wavelet basis selection. It is expected that these enhancements would further enhance the performance, although they would also significantly increase the complexity. Finally, we note that there is nothing wavelet-specific about this algorithm, and that still image and video coding algorithms based on the block DCT could also be improved by incorporation of the techniques described here.

IV. Acknowledgements

The authors wish to thank Weixing Zhang of Washington State University for providing the running time comparison given in Section III. The authors also thank the reviewers for helpful

comments and suggestions, and in particular for providing PSNR numbers for the Barbara image for the Said and Pearlman algorithm in the final row of Table 1.

References

[1] M. Vetterli and C. Herley, "Wavelets and filter banks: Theory and design," *IEEE Trans. on Signal Proc.*, vol. 40, pp. 2207-2232, 1992.

[2] J. Villasenor, B. Belzer, and J. Liao, "Wavelet Filter Evaluation For Image Compression", *IEEE Transactions on Image Processing*, vol. 4, pp. 1030-1060, August, 1995.

[3] K. Ramchandran and M. Vetterli, "Best wavelet packet bases in a rate-distortion sense," *IEEE Transactions on Image Processing*, vol. 2, pp. 160-175, February 1993.

[4] Z. Xiong, C. Herley, K. Ramchandran, and M.T. Orchard, "Space-frequency quantization for a space-varying wavelet packet image coder," *Proc. IEEE Int. Conf. on Image Proc.,* vol. I, pp. 614-617, October 1995.

[5] J. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients", *IEEE Tran. on Signal Processing,* vol. 41, no. 12, pp. 3445-3462, December, 1993.

[6] Z. Xiong, K. Ramchandran, and M.T. Orchard, "Joint optimization of scalar and tree-structured quantization of wavelet image decompositions," *Proc. 27th Annual Asilomar Conf. on Sig., Syst., and Comp.,* Pacific Grove, CA, November 1993.

[7] A. Said and W.A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," to appear in *IEEE Trans. on Circuits and Systems for Video Technology*, 1996.

[8] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic Coding for Data Compression", *Comm. ACM*, vol. 30, pp. 520-540, June 1987.

[9] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. on Image Proc.*, vol. 1, pp. 205-220, 1992.

[10] Weixing Zhang, Washington State University, personal correspondence.