The Compression Effects of the Binary Tree Overlapping Method on Digital Imagery

LOUIS J. DIMENTO AND SIMON Y. BERKOVICH

Abstract—**This paper describes a new method of digital image compression called binary tree overlapping. With this method, an image is divided into bit planes that are transformed into a binary tree representation in which initial identical portions of different lines in the bit planes are transmitted as one path in the tree. To increase compression, distortion can be introduced by considering two lines which differ in less than a preset number of bit positions to be identical. A time efficient method of implementing binary tree overlapping based on a communication technique called content-induced transaction overlap is outlined. With this technique, only simple, logical operations are needed and the compression time is proportional to the number of bits in the compressed image. Simulation studies indicate that binary tree overlapping produces good quality images with a compression ratio of about 3. In terms of its implementation and compression efficiencies, BTO is comparable to first-order DPCM.**

I. Introduction

WHILE digital imagery has many advantages over analog, the amount of data required for the digital representation is great. As a result, considerable effort has been devoted to digital image compressioin. Many of the techniques which have been proposed fall into one of two major categories [10], predictive coding, of which differential pulse code modulation (DPCM) [6], [13] is an important example, and transform coding, particularly the cosine transform [2]. Interpolative coding is another important category, but it appears to be inferior to adaptive versions of the first two [12]. Other methods, such as entropy coding [9], are often used in combination with some of the techniques indicated above. In addition, as hardware costs decrease, vector quantization [8] is becoming more practical.

This paper describes a new compression method called binary tree overlapping. With binary tree overlapping (BTO), an image is divided into bit planes, each of which contains horizontal lines of bits. The lines are transformed into a binary tree representation in which identical portions of different lines are transmitted as one path in the tree. BTO is distortionless, although compression can be increased by assuming that similar lines are identical and treating them as one path in the tree. It can be applied to a sequence of video images as well as one still image.

In the following section, the binary tree representation and the technique for introducing distortion are described. In Section III an expression for the compression factor is derived. A description of an efficient hardware implementation is presented in Section IV. This is followed by a discussion of error control techniques in Section V. Section VI presents the results of simulation studies assessing the objective and subjective performance of BTO. The implementation and compression efficiency of BTO are compared to first-order DPCM, which is easy to implement and fast, and to the cosine transform, which requires considerable computation but achieves greater compression efficiency. The conclusions constitute Section VII, the final section.

Paper approved by the Editor for Image Processing of the IEEE Communications Society. Manuscript received August 20, 1987; revised May 18, 1988. This work was supported in part by a grant from Allied-Signal Corporation.

L. J. DiMento is with AT & T Bell Laboratories, Middletown, NJ 07748.

S. Y. Berkovich is with Allied-Signal Aerospace Technology Center, Columbia, MD 21045.

II. Binary Tree Representation and Distortion Technique

Binary tree compression is illustrated in Fig. 1. The idea is to represent a collection of frame lines by a binary tree in which each frame line to be transmitted is a path from the root to a leaf. A left branch in the tree can represent a 0 in the line and a right branch can represent a 1. Compression is achieved because frame lines share the same initial path as long as their initial bits are the same. Thus, only bits from the point where a line differs from the previous line together with the location of this bifurcation point need be transmitted. The portion of a line which is transmitted is called a fragment. The fragments are transmitted from left to right in the tree, that is, in lexicographic order rather than in the original order. Therefore, the location of a line in a frame is identified by appending the lines with tags, which are also included in the binary tree representation. An image is restored from the binary tree by reconstructing the lines based on the paths from the root to a leaf. The tag which completes a path indicates the location of a line in the image and the bitplane it is in.

The first part of Fig. 1 shows a collection of 4 frame lines, each one followed by a 2 bit tag. The second part shows the binary tree representation of these liens. Since there are four lines, the tree contains four leaves. Each line can be determined by following a particular path from the root to a leaf with a left branch representing 0 and a right branch representing 1.

In general, the binary tree is represented as the sequence that is obtained by performing a depth-first transversal of the tree and writing 0 when going left and 1 when going right. (A depth-first traversal is a systematic passage over all the nodes in the tree starting from the root and proceeding toward the leaves [2].) The sequence of zeros and ones is interspersed with pointers to the bifurcation points, called bit competitions because of the nature of the hardware implementation, discussed below. All paths in the binary tree have the same length, corresponding to the left of a frame line (l) plus tag (t). The bit competitions have lengths about $\log(l + t)$. Therefore, the tree can be transmitted by alternating between variable length line fragments and numbers indicating how far up the path the bifurcation for the next fragment is. Since all paths have the same length, this is also the length of the next fragment. In Fig. 1, the sequence SENT shows the representation of the binary tree that is transmitted and RECONSTRUCTED presents the restored information. It can be seen that a 1 always follows a bifurcation point and therefore need not be transmitted.

Although binary tree compression exploits some of the interline redundancies, if two lines differ in one bit all subsequent similarities are ignored. Thus compression may be lost for lines in which beginning bits differ but subsequent bits are the same. This limitation can be alleviated by allowing some mismatches in lines. That is, if the number of mismatches between two lines are below some prespecified level, the lines are assumed to be identical and only one fragment is transmitted. Tolerating mismatches in this way introduces distortion into the reproduced image.

0090-6778/90/0800-1260$01.00 © 1990 IEEE

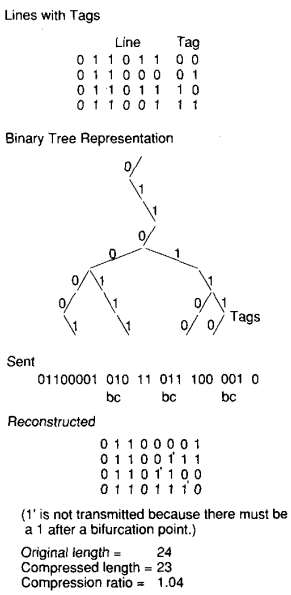


Fig. 1. Example of binary tree representation and transformation.

The time required to perform binary tree overlapping with mismatch tolerance on a computer is proportional to ln^2 where l is the line length and n is the number of lines. It is much faster when implemented with a special purpose device which performs a process similar to content-induced transaction overlap (CITO), an approach to multiaccess communications in which control is induced by the messages' contents [4]. It is shown below that, using this implementation, the time required is proportional to the number of bits in the compressed representation.

BTO can easily be applied to video imagery, since the lines which are transformed to a binary tree can come from a sequence of frames as well as one still image.

III. Theoretical Expression for Compressed Ratio

Estimates of the amount of compression that can be obtained with the BTO method can be made by using the general relation among the nodes with zero, one and two children in a binary tree. If the number of such nodes are n_0, n_1, and n_2, respectively, then it is easily shown that this relation is

$$n_0 = n_2 + 1. \tag{1}$$

The number of bits required for the binary tree representation of the video imagery, is the number of bits in the traversal B plus the number required for the bit competition. The number of bits in the traversal portion is equal to the total number of branches in the tree minus n_2. The reason for subtracting n_2 is that there are n_2 bit competitions (one for every bifurcation point in the tree) and, as noted in the previous section, the first bit after every bit competition is always 1 and therefore does not have to be transmitted. Thus the number of bits N required for the BTO representation is

$$N = B + bn_2$$

where b is the number of bits required for each bit competition.

The total number of frame lines equals the number of bitplanes times the vertical dimension of the image. Letting this total be denoted by n, and noting from (1) above that $n_2 = n_0 - 1$ and that n_0 equals n gives

$$N = B + b(n - 1). \tag{2}$$

Since the original number of bits is ln, this gives a general compression ratio k of

$$k = \frac{ln}{B + b(n - 1)}. \tag{3}$$

The values of b must be great enough to indicate any point in the tree; that is, there must be enough bits to specify any value up to the depth of the tree, which equals the length of a frame line plus tag $(l + t)$. If n is the total number of frame lines (and the lines are numbered from 0 to $n - 1$) then the tag length is given by

$$t = \lfloor \log (n - 1) \rfloor + 1$$

and the length of each bit competition equals

$$b = \lfloor \log (l + t - 1) \rfloor + 1.$$

If these equations are substituted into (3), everything except B is expressed in terms of the image characteristics. Thus the compression ratio k is given by

$$k = \frac{ln}{B + (\lfloor \log (l + \lfloor \log (n - 1) \rfloor) \rfloor + 1)(n - 1)}. \tag{4}$$

A simple approximation to the compression ratio can be obtained by replacing B with n times the average fragment length a eliminating the floors, replacing $n - 1$ with n, and ignoring $\log n$ which is small compared with l. This gives

$$k = \frac{l}{a + \log l}. \tag{5}$$

Maximum compression is achieved if a approaches zero and is given by

$$k_{max} = \frac{l}{\log l}.$$

For example, for a 256 by 256 image, $k_{max} \approx 30$.

IV. Hardware Implementation of Binary Tree Overlapping

This section outlines a hardware implementation of BTO which is based on the content-induced transaction overlap (CITO) communications technique [4]. With this implementation a frame is loaded into a bit serial associative memory, one line per word in memory. Each bit position of all words of the memory is simultaneously addressable, but only one bit of a word can be accessed at a time. This implementation is a specific type of associative processor [17] and has an advantage of fast operation.

Fig. 2 illustrates a CITO device,which includes a line-tag component and a bit position (BP) component. The line portion of the line-tag component is loaded with a frame, and the tag portion permanently contains tags identifying each line. The number of bits of a line which remain to be transmitted is called its bit position since it indicates the position of the next bit to be transmitted. This is the number of branches from the bifurcation point to the leaf on the binary tree. Each location in the BP component contains the bit position of the corresponding line, measured from right to left. Each line also has associated with it an activation register (ac) which is 1 when a line is transmitting and 0 otherwise.

The CITO technique is applied to each bit in all frame lines simultaneously by scanning the lines starting from the point where the next line to be transmitted bifurcated from the previous line. The scan goes from left to right and covers both the line and tag component of memory. Each line's BP register is decremented until there is a conflict, which occurs if its bit is 1 and the corresponding bit for any other active line is 0.

When a conflict occurs for a particular line, it is deactivated (ac = 0) and it begins counting the number of mismatches with active lines. When the scan reaches the end of the line and the

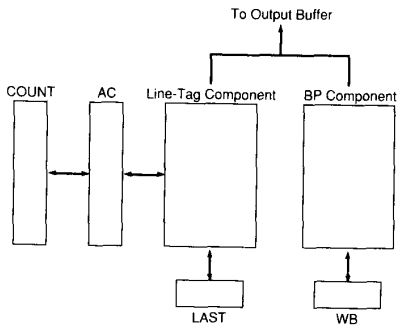


Fig. 2. An individual CITO device.

beginning of the tag, the number of mismatches may be below the preset tolerated value. If so, the line is reactivated and its BP register is set to t (the tag length).

When the scan reaches the end of the tag, each line's BP register will indicate the bit position where it finally became inactive. Any line whose BP register contains 0 is finished. The other lines participate in bit competition, wherein the BP memory is scanned and the result, which is the smallest nonzero value in any BP register is read into the word boundary register (WB). This register contains the length of the next fragment and is so called because it indicates the end (boundary) of a line. All lines whose BP register equals its WB register are activated and the next scan of the line-tag memory begins at the position indicated in WB.

The register, LAST, contains a copy of the previous line which was transmitted. The COUNT register of each line contains the number of mismatches with this line. During a scan this register is decremented for each bit in a line which is the same as the bit in the line currently being sent but different from that in LAST. It is incremented if the opposite is true, and it is not changed if it is either the same as or different from both. Thus, the number of mismatches with the current line can be determined even though the scan may not start at the beginning of memory.

The time T required for this algorithm is

$$T = (n+1)a + (n-1)b \approx n(a + \log(l + \log n)).$$

As noted in Section III, a is the average fragment length and b is the length of the bit competition. These are $n+1$ scans because an extra one is needed to initialize the register LAST. The time required is about proportional to the number of bits in the compressed representation. In comparison, DPCM requires m^2 additions $+m^2$ multiplications for an m by m image and the fast cosine transform [5], [18] requires $1.5m^2 \log N$ additions $+m^2 \log N$ multiplications for the image when each block contains N pixels. For example, using BTO (with a compression ratio of 4) requires only about 0.12 million logical operations for a 256 by 256 image. Meanwhile, DPCM requires 0.05 million additions +0.05 millions multiplications while the 16 × 16 DCT requires 0.75 million additions +0.5 million multiplications. Therefore, if multiplication is performed by adding and shifting, BTO can be faster than DPCM and DCT.

V. Error Control

With binary tree compression, errors can occur either in a fragment of the binary tree or in the bit competition. If a bit changes from 0 to 1 or vice versa in a fragment, this bit will be incorrect in all lines which are constructed from fragments that bifurcate from the same path after the error. Such an error will thus result in a number of incorrect pixel values at certain points along a vertical line in a portion of the image.

An error in a bit competition is more serious than a fragment error because it can cause loss of synchronization. Such an error causes the receiver to expect the wrong number of bits in the following fragment. As a result, the next bit competition will not be correctly located and the error will propagate to the subsequent fragments.

Three error control techniques can be used to decrease the likelihood of an error and to limit propagation when an error does occur. The first is to use run length encoding on the fragments with one code set aside as a special end-of-line marker. As noted above, all fragments except the first start with a one. Although it is not necessary to transmit this one, if run-length encoding is used it is advantageous to include it because this simplifies transmission of only the run lengths and not the values. The run-length code can then consist of a series of fixed length "nibbles," and a code can be set aside as a special end-of-line indicator. Second, because of their importance, the bit competitions can be error control coded.

The third measure for error control is to introduce forced updating by regularly transmitting an entire line instead of a fragment in the binary tree. This will prevent the propagation of any error that may have occurred since the previous update. To reestablish synchronization in case an error causes the receiver to count an incorrect number of fragments, another special code can be set aside to indicate that forced updating follows. It is also important for the receiver to know when the end of a tree occurs. In the absence of error the receiver can determine this simply by counting the fragments. However, since an error can result in an incorrect count, a third special code can be used to indicate the end of the tree.

If the receiver observes a discrepancy between the fragment length indicated in the bit competition and that derived from the run lengths, it assumes the bit competition is correct. This is reasonable when the bit competitions are error control coded because the probability of error is reduced.

The introduction of run-length coding changes the nature of a fragment error somewhat. An error in a run length code, results in an incorrect run length. As in the pure binary tree technique, such an error will propagate to lines whose fragments bifurcate below it. However, in this case an error will result in the shifting of a number of lines in the bit planes constituting the image. As with a bit competition error, this error propagation is arrested by forcing updating.

VI. Simulation Studies

The performance of binary tree compression was evaluated by simulations with a number of different pictures, including computer created binary images and multilevel images which were digitized from film. The simulation was extended to standard images such as the Girl picture shown in Fig. 4 and to a sequence of video images. The results for a sequence of images were found to be about the same as those for single images, which are provided below.

BTO was tested in its simple form and in combination with run-length encoding of the fragments. Bit plane analysis was also performed to determine the effects of each bit plane on compression. The entropy of the individual bit planes assuming a first-order Markov source was measured. Although this assumption ignores interline and interframe redundancies, the results are useful for comparing the randomness of the bit planes.

When mismatch tolerance equals zero, the simple compression ratio and the run-length encoded compression ratio for the Girl picture and 1.1 and 0.8, respectively. Table I presents these ratios for each bit plane along with their entropies assuming a first-order Markov process. The least significant bit plane is the first row in the table.

These results indicate that the distortionless version of BTO achieves limited compression. The bit plane analysis indicates that the major reason for this is the high degree of randomness in the lower bit planes, which the entropy figures attest to. The least significant bit plane has very high entropy as expected, while the entropy of the most significant plane is quite low. Although, as noted above, these entropies do not provide a lower bound to binary tree compression, they do clearly indicate the high variability of the least significant planes. Thus, the fact that no compression is achieved in the low bit planes is not surprising. However, the

TABLE I
ENTROPY AND COMPRESSION RATIOS AS A FUNCTION OF BIT PLANE

bitplane	entropy	simple compression ratio	compression ratio using r.l.e.
1	.970	.967	.568
2	.990	.966	.523
3	.987	.967	.547
4	.946	.974	.643
5	.808	1.000	.870
6	.590	1.123	1.350
7	.319	1.514	2.349
8	.137	2.904	4.454

TABLE II
DISTORTION THRESHOLDS

bitplane	distortion threshold
1	80
2	80
3	70
4	50
5	30
6	10
7	1
8	1

compression ratio for the most significant bit plane is nearly three, indicating that substantial compression may be achieved if mismatch tolerance is concentrated in the low-order bit planes.

Table I also demonstrates that, when run-length encoding is included, the compression ratio decreases for high entropy planes and increases for low entropy ones. This reflects a general trend: run-length encoding tends to make a low compression ratio even lower, but it improves a high ratio. This is true because, as interline redundancy increases, intraline redundancy also tends to increase.

When mismatch tolerance is introduced, some distortion is introduced into the picture and the compression ratio is increased. The measure of distortion is the normalized mean square error, that is, the ratio of the mean square error to the variance of the pixel values.

As the above bit plane analysis indicates, in order to maximize the effectiveness of mismatch tolerance, the number of mismatches must be allowed to vary with the bit plane. To determine the approximate number of mismatches that should be tolerated in each bit plane, the error rate as a function of mismatch tolerance was examined for each of the bit planes in a number of pictures. The error rate is the average number of bits per line that are changed as a result of mismatch tolerance. The results were that for bit planes 1 and 2 (the least significant), the error rate was zero when the number of allowed mismatches was under 80. At this level of mismatch tolerance, errors began to occur, indicating that the distortion threshold is about 80 for bit planes 1 and 2. Above the threshold, the error rate gradually increased to a maximum of about 50% at about 150 mismatches. For bit planes 3–6, the thresholds and the maximum error rate declined, and for bit planes 7 and 8 the thresholds were both just one mismatch, and the maximum rate rose to about 20%. These results are approximate, although the variation from picture to picture is small. The distortion thresholds are summarized in Table II. The conclusion is that the level of mismatch tolerance should be large for the least significant bit plane and gradually decrease to a few mismatches for the most significant bit plane.

The number of bits per pixel and the normalized mean square error were obtained for a variety of levels of mismatch tolerance for the images examined. The results for binary tree overlapping are compared with the rate distortion bound [16], [3] and with other data compression techniques. Fig. 2 presents the rate distortion bound, the simulation results for binary tree overlapping on the Girl picture and the theoretical bit rate for the 16×16, optimally quantized cosine transform and for first-order DPCM (both nonadaptive and entropy coded) [14]. The BTO results for the other images were similar.

The following assumptions were used in deriving the rate distortion bound and the theoretical bit rates for DPCM and transform coding. The images come from a stationary, Gaussian source with separable Markovian correlations. The row and column correlation coefficients are both equal to 0.95. The rate distortion bound for such a source is given by the following expression [7], under the assumption that distortion is low:

$$R(D) = \frac{1}{2} \log \frac{\left(1 - \rho^2\right)^2 \sigma^2}{D}.$$

In calculating the performance of binary tree overlapping, an estimate of the variance was obtained by calculating the ensemble average for a number of pictures. The resulting estimate is 5500. For purposes of comparison, the row and column correlations were also obtained using ensemble averages, and the results are consistent with the values given above.

The results shown in Fig. 3 indicate that BTO is about 3.5 bits per pixel above the rate distortion bound, about 3 bits above the cosine transform and about 1.5 bits above DPCM for a given normalized mean square error. The other images tested produced similar results.

Although mean square error is widely used as a measure of distortion, it is generally not linearly related to the subjective judgment of image distortion. It is possible that a image with a small mean square error will appear more objectionable than one with a larger error. To evaluate the subjective performance of binary tree overlapping, a number of images were reconstructed after applying compression. The results for the Girl and House images are presented below.

In both cases four images are presented: the original image and three reconstructed images. These three images have been subjectively rated by the author according to the rating system used by the mean opinion score (MOS) technique [11]. According to this system, pictures are rated as excellent (impairment is imperceptible), good (perceptible but not annoying), fair (slightly annoying), poor (annoying), and unsatisfactory (very annoying). The pictures shown in Figs. 4–7 for the Girl image and Figs. 8–11 for the House image indicate that an excellent picture can be attained when the compression ratio is about 2, a good image when the ratio is 3 and a fair image when the ratio is 4. Also included in these figures are the mismatch tolerances used for each picture. Table III summarizes the results for the Girl and House pictures and a third image, Tree.

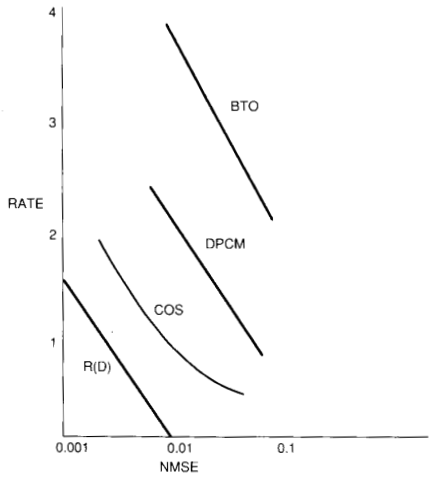
These three images differ substantially in terms of diversity, with Tree being the most diverse and House the least. Despite these differences, the results in Table III are fairly consistent, reinforcing the conclusion that excellent, good and fair results can be achieved with compression ratios of about 2, 3, and 4, respectively.

A comparison of the objective and subjective results indicates that the performance of BTO improves when evaluated subjectively. For example, it is inferior to DPCM when evaluated on the basis of mean square error, but a subjective comparison indicates that the two methods perform about the same. Also, because of the nature of the distortion introduced by BTO, the reconstructed image may be improved by passing a low-pass mask over it. One reason that BTO fares better subjectively is that the error introduced by mismatch tolerance tends to be randomly distributed throughout the image. Humans tend to be tolerant of snow-like noise, a factor which the concept of dithering [15] exploits.

VII. CONCLUSION

Binary tree overlapping using the CITO associative processing technique is time efficient: only logical operations are required and the number of operations is proportional to the number of bits in the compressed image. The hardware consists of a bit-serial associative processor and an output buffer. Error control techniques can effectively be used to decrease the chance of error and arrest its propagation without substantially reducing compression.

While BTO is distortionless, compression can be increased by introducing mismatch tolerance. Simulation studies show that substantial compression without excessive distortion can be achieved by



Binary tree overlapping on the Girl picture, the 16 by 16 cosine transform and DPCM compared with the rate distortion bound.

Fig. 3. Bit rate versus normalized mean square error.

Fig. 4. Original Girl picture.

Fig. 5. Girl picture with low mismatch tolerance. Compression ratio = 2.0.

Fig. 6. Girl picture with moderate mismatch tolerance. Compression ratio = 3.0.

Fig. 7. Girl picture with high mismatch tolerance. Compression ratio = 4.78

Fig. 8. Original House picture.

Fig. 9. House picture with low mismatch tolerance. Compression ratio = 1.75.

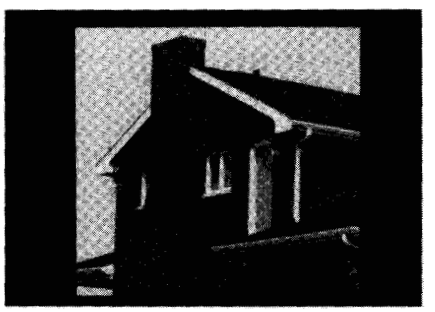


Fig. 10. House picture with moderate mismatch tolerance. Compression ratio = 2.75.

varying the level of mismatch tolerance with bit plane. An excellent quality image is produced with a compression ratio of 2, a good image with a ratio of about 3 and a fair image with the ratio of about 4. Similar results are achieved for single images and sequences of images.

Because the BTO technique tends to exploit different redundancies than existing methods, it can be used in combination with methods such as transform coding and vector quantization. It would

U-M-I

Due to a lack of contrast this page did not reproduce well.

Fig. 11. House picture with high mismatch tolerance. Compression ratio = 4.05.

TABLE III
IMAGE QUALITY, COMPRESSION RATIOS AND NORMALIZED MEAN SQUARE ERROR

Image		Girl	House	Tree
Quality				
Excellent	(a)	2.00	1.75	1.81
	(b)	1.85	1.53	1.62
	(c)	.01	.002	.006
Good	(a)	3.00	2.75	2.59
	(b)	3.32	3.35	2.71
	(c)	.055	.024	.038
Poor	(a)	4.78	4.05	4.80
to	(b)	5.34	5.27	5.32
Fair	(c)	.176	.089	.179

also be particularly beneficial in an associative processing environment where it could produce compression without additional special hardware.

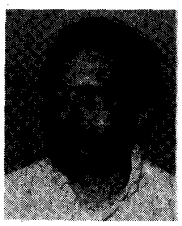
ACKNOWLEDGMENT

The authors would like to thank P. Keating and C. Walter for their support, as well as C. R. Wilson for providing valuable comments and for performing preliminary simulation studies. The authors would also like to thank H. Helgert, M. Loew, A. Meltzer, R. Pickholtz, and J. Sibert for their helpful suggestions.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman *Data Structures and Algorithms*. Reading, MA: Addison-Wesley, 1983.
[2] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, pp. 90-93, Jan. 1974.
[3] T. Berger, *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Englewood Cliffs, NJ: Prentice-Hall, 1971.
[4] S. Y. Berkovich and C. R. Wilson, "A computer communication technique using content-induced transaction overlap," *ACM Trans. Comput. Syst.*, vol. 2, no. 1, pp. 60-77, Feb. 1984.
[5] W. Chen, C. H. Smith, and C. S. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans.Commun.*, vol. COM-25, pp. 1004-1009, Sept. 1977.
[6] C. C. Cutler, Differential quantization of communication signals, Pat. 2-605-361, Application June 1950, Issuance July, 1952.
[7] L. D. Davisson, "Rate-distortion theory and application," *Proc. IEEE*, vol. 60, pp. 800-808, July 1972.
[8] R. M. Gray, "Vector quantization," *IEEE ASSP Mag.*, pp. 4-28, Apr. 1984.
[9] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, pp. 1098-1101, Sept. 1952.
[10] A. K. Jain, "Image data compression: A review," *Proc. IEEE*, vol. 69, pp. 349-389, Mar. 1981.
[11] N. S. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
[12] A. N. Netravali and J. O. Limb, "Picture coding: A review," *Proc. IEEE*, vol. 68, pp. 366-406, Mar. 1980.
[13] J. B. O'Neal, "Predictive quantizing systems (differential pulse code modulation) for the transmission of television signals," *Bell Syst. Tech. J.*, vol. 45, no. 5, pp. 689-721, 1966.
[14] W. K. Pratt, *Digital Image Processing*. New York: Wiley, 1978.
[15] L. G. Roberts, "Picture coding using pseudo-random noise," *IRE Trans. Inform. Theory*, vol. 8, pp. 145-154, Feb. 1962.
[16] C. E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," *IRE Nat. Convention Rec. 1959*, Part 4, pp. 142-163.
[17] K. J. Thurber, *Large Scale Computer Architecture, Parallel and Associative Processors*. Rochelle Park, NJ: Hayden, 1976.
[18] Z. Wang, "Reconsideration of 'A fast computational algorithm for the discrete cosine transform'," *IEEE Trans. Commun.*, vol. 31, pp. 121-123, Jan. 1983.

★



Louis J. DiMento received the B.S. degree in bioengineering from Syracuse University in 1966, the M.S. degree in systems engineering from the University of Michigan in 1968, and the Ph.D. degree in computer science from George Washington University in 1987.

He was a visiting Assistant Professor in Computer Science at American University in Washington, DC, for the 1987-1988 academic year. In 1988 he joined AT & T Bell Laboratories in Middletown, NJ, where he is currently developing local area network software.

Dr. DiMento's research interests include network protocols, distributed operating systems, and computer organization. He is a member of the IEEE Communications Society and the ACM.

★

Simon Y. Berkovich is a Professor of Engineering and Applied Science at the George Washington University. He also works part-time for Allied-Signal Aerospace Technology Center. His research interests include information systems, computer organization, and mathematical modeling.

Dr. Berkovich is a member of the IEEE Computer Society, ACM, IMACS, and American Physical Society.