[4] V. K. Prabhu, "MSK and offset QPSK modulation with bandlimiting filters," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-17, Jan. 1981.
[5] S. A. Rhodes, "Effects of hardlimiting on bandlimited transmissions with conventional and offset QPSK modulation," in *Proc. IEEE Nat. Telecommun. Conf.*, 1972, pp. 20F/1–20F/7.
[6] G. Satoh and T. Mizuno, "Nonlinear satellite channel design for QPSK/TDMA transmission," presented at the 5th Int. Conf. Digital Satellite Commun., Genoa, Italy, Mar. 23–26, 1981.
[7] J. Huang, K. Feher, and M. Gendron, "Techniques to generate ISI and jitter-free bandlimited Nyquist signals and a method to analyze jitter effects," *IEEE Trans. Commun.*, vol. COM-27, Nov. 1979.
[8] F. Jager and C. B. Dekker, "Tamed frequency modulation, a novel method to achieve spectrum economy in digital transmission," *IEEE Trans. Commun.*, vol. COM-26, May 1978.
[9] T. Le-Ngoc, K. Feher, and H. Pham Van, "Power and bandwidth efficient IJF quadrature modulation techniques for linear and nonlinear channels," in "Regenerative transponders for more efficient digital satellite systems," CRC Rep. OSU80-00197.
[10] Intelsat TDMA/DSI System Specification BG-42-65, May 1981.
[11] T. Le-Ngoc and K. Feher, "Performance of IJF-OQPSK modulation schemes in the presence of noise, interchannel and cochannel interference," in *Proc. IEEE Nat. Telecommun. Conf.*, New Orleans, LA, Nov. 1981.
[12] R. Lucky, J. Salz, and E. Weldon, *Principles of Data Communication*. New York: McGraw-Hill, 1968.
[13] K. Feher, *Digital Communications: Microwave Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1980.
[14] T. Le-Ngoc, K. Feher, and H. Pham-Van, "New modulation techniques for low cost power and bandwidth efficient satellite earth stations," *IEEE Trans. Commun.*, vol. COM-30, Jan. 1982.
[15] M. C. Austin and M. U. Chang, "Quadrature overlapped raised cosine modulation," *IEEE Trans. Commun.*, vol. COM-29, Mar. 1981.
[16] W. Foster and M. Chang, "Performance of several pulse shaping techniques in nonlinear TDMA environment," in *Proc. IEEE Int. Conf. Commun.*, Denver, CO, June 1981.
[17] D. Divsalar and M. K. Simon, "Performance of quadrature overlapped raised cosine modulation over nonlinear satellite channels," in *Proc. IEEE Int. Conf. Commun.*, Denver, CO, June 1981.
[18] K. Feher, "Filter," U.S. Patent 4 339 724, issued July 13, 1982; Canada Disclosure 327 365, filed May 10, 1979.
[19] ——, *Digital Modulation Techniques in an Interference Environment*. Gainesville, VA: Don White Consultants, 1977.
[20] ——, *Digital Communications: Satellite/Earth Station Engineering*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

Digital Image Compression by Outer Product Expansion

DIANNE P. O'LEARY AND SHMUEL PELEG

***Abstract*—We approximate a digital image as a sum of outer products dxy^T where d is a real number but the vectors x and y have elements +1, −1, or 0 only. The expansion gives a least squares approximation. Work is proportional to the number of pixels; reconstruction involves only additions.**

Paper approved by the Editor for Signal Processing and Communication Electronics of the IEEE Communications Society for publication without oral presentation. Manuscript received October 20, 1981; revised September 23, 1982. This work was supported by the Office of Naval Research under Grant N000014-76-C-0391, by the U.S.-Israel Binational Science Foundation, and by the National Science Foundation under Grant MCS-79-23422.

D. P. O'Leary is with the Department of Computer Science, University of Maryland, College Park, MD 20742.

S. Peleg was with the Computer Science Center, University of Maryland, College Park, MD 20742, on leave from the Department of Mathematics, Ben-Gurion Unversity, Beersheva, Israel. He is now with the Department of Computer Science, Hebrew University, 91904 Jerusalem, Israel.

INTRODUCTION

Consider the problem of digital image compression: given a gray level image A of dimension $n \times n$ with β bits for the gray level of each of the n^2 pixels, store an approximation to A in less than βn^2 bits. The survey papers by Jain [4] and Thomas [7] discuss the many techniques that have been proposed for data compression. The technique proposed in this paper is a "transform technique," where here the transform is image dependent. We approximate the image by an outer-product expansion:

$$A \approx \sum_{k=1}^{K} d_k x_k y_k^T$$

where x_k and y_k are $n \times 1$ vectors with elements equal to +1, −1, or 0, and the d_k are real constants.

Previous outer product expansions have been of two types.

1) The vectors x_k and/or y_k are predetermined (for example, Hadamard-Walsh vectors or Haar vectors). Generally, K needs to be rather large in order to approximate A well. However, there is no need to store the vectors. The work in computing each d_k is linear in the number of pixels.

2) The vectors x_k and y_k are image dependent but are vectors of real numbers. They are usually generated using the singular value decomposition (SVD) [1]–[3], a matrix factorization technique. These are the optimal x_k, y_k, and d_k in the sense of minimizing the sum of squares of deviations between the original and the approximate pixel values, i.e.,

$$\sum_{i,j=1}^{n} \left(a_{ij} - \sum_{k=1}^{K} d_k x_{ik} y_{jk} \right)^2 \text{ is minimized}$$

where x_{ik} and y_{jk} are components of the vectors x_k and y_k. The total work is proportional to the number of pixels to the 3/2 power.

A method of Moler and Stewart [5] approximates the results of the second algorithm, but in less work. This method still stores $2K$ real n-vectors.

The method proposed in this paper uses the same error criterion as the SVD algorithm, but restricts the elements of vectors x_k and y_k to +1, −1, and 0 only. Once y_k (or x_k) is chosen, then x_k (or y_k) and d_k minimize the resulting sum of squares of deviations between the current residual pixel values and the values in the outer product $d_k x_k y_k^T$. The work per outer product is proportional to the number of pixels.

II. THE ALGORITHM

The idea is as follows.

For $k = 1, \cdots, K$

1) Choose a vector y.

2.1) Given the current y, compute the vector x which solves the problem

$$\min_{d,x} r(x, y, d)$$

$$\text{where } r(x, y, d) = \sum_{i,j=1}^{n} (a_{ij}^{(c)} - d x_i y_j)^2$$

and $A^{(c)}$ is the current residual picture.

0090-6778/83/0300-0441$01.00

2.2) Given the current x, compute the optimal y and d to solve the problem $\min_{d,y} r(x, y, d)$.

2.3) Repeat steps 2.1 and 2.2 until the improvement in deviation is less than a given threshold.

3) Prepare for the next iteration: the current values of x, y, and d become x_k, y_k, and d_k, and the current picture is updated as

$$A^{(c)} = A^{(c)} - d_k x_k y_k{}^T.$$

In detail, the algorithm is as follows.

One is given the original digitized picture A, and the number K of outer products to be computed.

Initialize the current image, $A^{(c)} = A$, and the residual

$$r = \sum_{i,j=1}^{n} a_{ij}{}^2.$$

For $k = 1, \cdots, K$

1) Initialize y. (See later notes.)

Initialize **change** to be 1, **improvement** to be 1.

2) Repeat steps 2.1-2.3 while **improvement** > 0.01.

2.1) Given the current y, find the optimal x using the following procedure:

Let

$$s_i = x_i \sum_{j=1}^{n} a_{ij}{}^{(c)} y_j$$

where $x_i = \pm 1$ is chosen so that $s_i \geq 0$, $i = 1, \cdots, n$.

Order the s_i's: $s_{i_1} \geq s_{i_2} \geq \cdots \geq s_{i_n}$.

Let

$$f_j = \left(\sum_{m=1}^{j} s_{i_m} \right)^2 \Big/ j, \quad j = 1, \cdots, n$$

and

$$f_J = \max_{j=1,\cdots,n} f_j.$$

Set $x_{i_{J+1}} = \cdots = x_{i_n} = 0$.

2.2) Given the current x, find the optimal y and c using the following procedure:

Let

$$s_i = y_i \sum_{j=1}^{n} a_{ji}{}^{(c)} x_j$$

where $y_i = \pm 1$ is chosen so that $s_i \geq 0$, $i = 1, \cdots, n$.

Order the s_i's: $s_{i_1} \geq s_{i_2} \geq \cdots \geq s_{i_n}$.

Let

$$f_j = \left(\sum_{m=1}^{j} s_{i_m} \right)^2 \Big/ j, \quad j = 1, \cdots, n$$

and

$$f_J = \max_{j=1,\cdots,n} f_j.$$

Set $y_{i_{J+1}} = \cdots = y_{i_n} = 0$, $s_{i_{J+1}} = \cdots = s_{i_n} = 0$,

and

$$d = \sum_{i=1}^{n} s_i \Big/ \left(J \sum_{j=1}^{n} |x_j| \right).$$

2.3) Find the change in the sum of squares of the deviations and the improvement:

$$\textbf{newchange} = d^2 J \sum_{j=1}^{n} |x_j|,$$

$$\textbf{improvement} = \frac{\textbf{newchange} - \textbf{change}}{\textbf{change}}, \text{ and}$$

$$\textbf{change} = \textbf{newchange}.$$

3) Set $x_k = x$, $y_k = y$, $d_k = d$, and update the current picture and residual.

$$A^{(c)} = A^{(c)} - d_k x_k y_k{}^T.$$

$$r = r - \textbf{change}.$$

Possible choices for the initial y vectors include the following.

1) The kth y could be the kth Hadamard vector, or a vector from another common basis set. In the experiments reported in this work, the Hadamard vectors were generated in order of increasing number of sign changes as in [6].

2) Pick the largest row of $A^{(c)}$ and define y by thresholding that row. (If all elements of the row have the same sign, then y should have only 0's and 1's, no -1's.)

3) At each iteration, choose y to be the vector of all 1's.

III. PROPERTIES OF THE ALGORITHM

1) Given any vector y, the algorithm finds the optimal choice for x and d, i.e., the choice which solves

$$\min_{x,d} \sum_{i,j=1}^{n} (a_{ij}{}^{(c)} - d x_i y_j)^2$$

where x is constrained to have elements $+1$, -1, or 0 only. The proof of this is given in the Appendix. Similarly, given a vector x, the algorithm finds the optimal y and d.

2) The loop in step 2 of the algorithm is guaranteed to terminate, since no iteration makes the residual larger; there are only a finite number of possible vectors x and y; and d is chosen optimally for each x-y pair. Thus, any tolerance greater than or equal to zero could be used in place of 0.01.

3) Because all of the x and y components are 1, 0, or -1, each step in the inner loop (2) requires only $2n + 2$ multiplications and $2n + 2$ divisions. Updating the current picture involves no multiplications or divisions.

4) Reconstructing the approximation later requires at most Kn^2 additions and subtractions, and no multiplications or divisions.

5) As an alternative, termination could be based on the size of $A^{(c)}$ rather than a fixed number K of outer products.

6) The cost in computation is linear in the number of pixels.

7) One disadvantage is that the algorithm is not guaranteed to ever reduce the residual image to zero, even though any

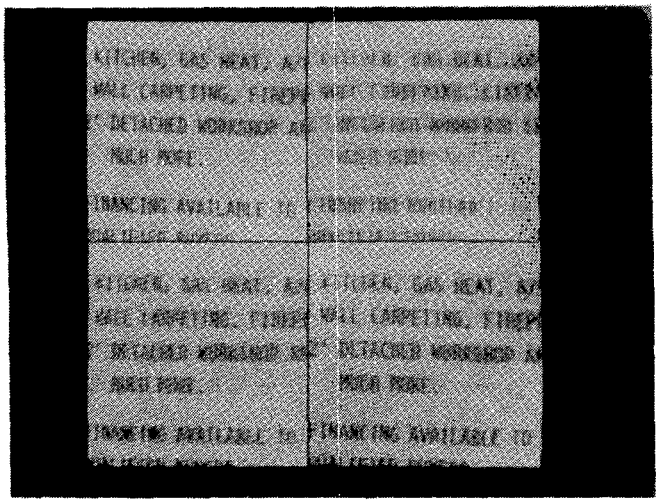

Fig. 1. Example 1: original and reconstructed pictures. Upper left: original picture. Upper right: reconstruction from 10 x-y pairs, 0.25 bits/pixel. Lower left: reconstruction from 30 x-y pairs, 0.75 bits/pixel. Lower right: reconstruction from 60 x-y pairs, 1.50 bits/pixel.

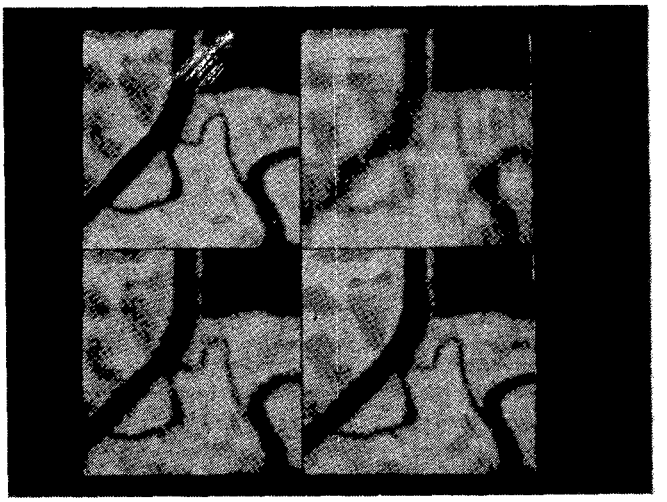

Fig. 2. Example 2: original and reconstructed pictures. As in Fig. 1.

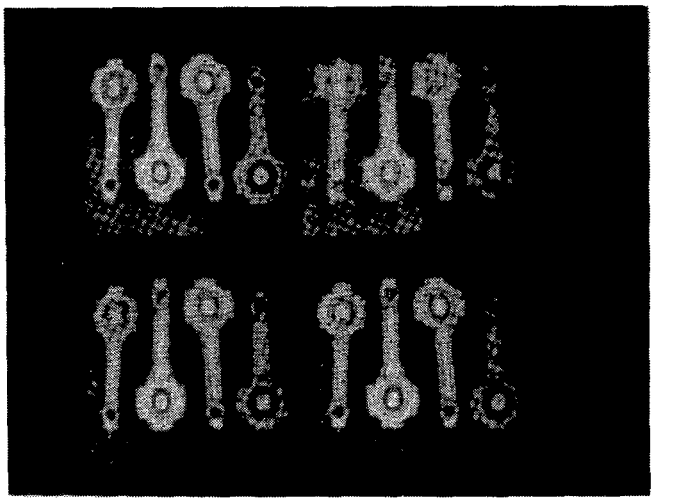

Fig. 3. Example 3: original and reconstructed pictures. As in Fig. 1.

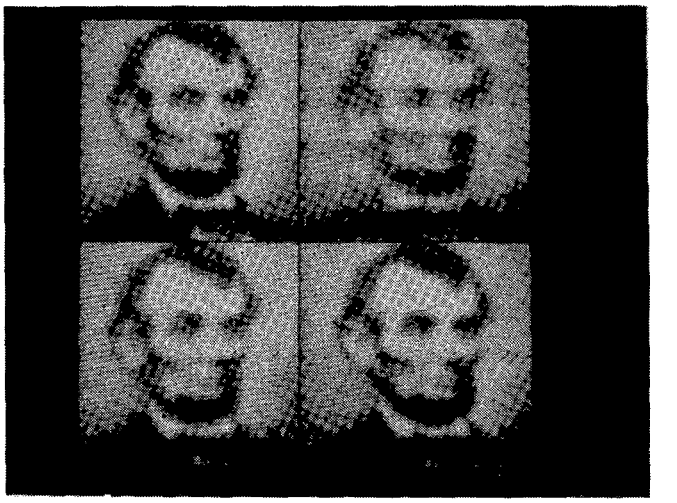

Fig. 4. Example 4: original and reconstructed pictures. As in Fig. 1.

image could be represented as the sum of at most n^2 of these outer products.

8) The processing can be arranged to access the image by rows only. Thus, the algorithm could be run with fast storage of a few vectors of length n, accessing the image sequentially from secondary storage.

9) Extension of the algorithm to three-dimensional images is possible.

IV. EXPERIMENTAL RESULTS

The algorithm was tested on several 64 X 64 and 127 X 127 images. Results for the larger images are shown in Figs. 1–4. The coefficient associated with each vector pair had 6 bit (one gray level) accuracy, as did the original pixel values. The initial y vectors were Hadamard vectors.

The 127 X 127 reconstructed images do preserve the major features of the original even at $K = 10$, and at $K = 30$ and 60 the detail improves. For pictures of size 64 X 64, about half as many vectors were needed.

The average error per pixel decreased rapidly at first and then more slowly. The average errors and average number of choices of y vectors per iteration are given in Table I.

Storage required per outer product is about $6 + 2 \cdot 127 \log_2 3$ or approximately 410 bits. Thus, the storage required for 10, 30, and 60 iterations is 0.25, 0.76, and 1.5 bits per pixel, respectively. Significant further compression could be achieved by run length encoding of the vectors. (See Fig. 5.)

Using an initial y vector of all 1's at each iteration, or using each Hadamard vector for two iterations instead of one, made little change in the pictures or errors but increased the number of inner iterations. For the vector of all 1's for the picture in Fig. 3, for example, the average number of y choices was 9.2.

The final error levels were limited by the truncation of the numbers d_j to integral gray levels. To obtain more accuracy, but still restrict arithmetic to 6 bit accuracy, the residual image can be scaled upward by a power of two whenever all elements have been reduced sufficiently, and the algorithm can be applied to this rescaled image.

V. CONCLUSIONS

We have presented an algorithm for image compression which (experimentally) achieves a storage rate of 1 bit per pixel without much loss of information.

TABLE I

Image	Average error per pixel k=0	k=10	k=30	k=60	Average number of y choices per iteration
Fig. 1	45.5	5.8	3.9	2.5	7.4
Fig. 2	33.3	6.3	4.0	2.6	7.7
Fig. 3	23.1	7.2	4.1	2.8	7.0
Fig. 4	33.7	4.8	2.6	1.7	7.7

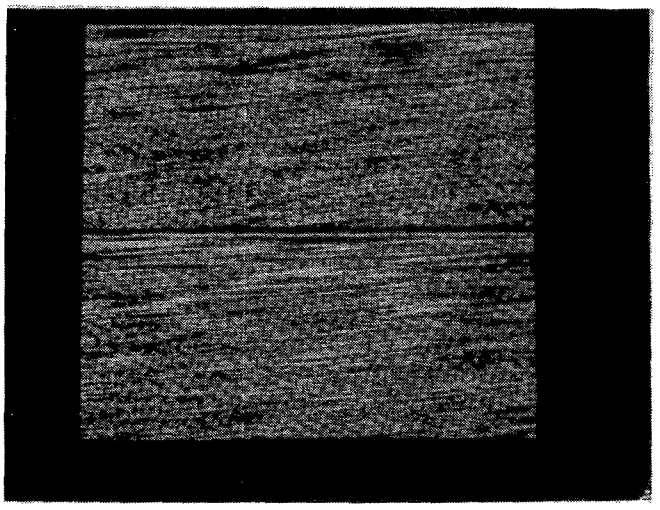

Fig. 5. Display of 60 x-y pairs for the picture of Fig. 4. -1, 0, and 1 are displayed as black, gray, and white, respectively. The upper row of the upper part is the first y and the upper row of the bottom part is the first x.

APPENDIX

PROOF OF OPTIMALITY

Here we show that the x computed in step 2.1 of the algorithm solves the problem

$$\min_{x,d} r(x, y, d)$$

$$\text{where } r(x, y, d) = \sum_{i,j=1}^{n} (a_{ij}^{(c)} - dx_i y_j)^2$$

$$x_i = +1, -1, \text{ or } 0.$$

This is a "mixed integer programming problem" with 3^n possible choices for the vector x, but we will show that the solution can be computed by examining only n of these possibilities. The argument is as follows.

Expanding the formula for $r(x, y, d)$ gives

$$r(x, y, d) = \sum_{i,j=1}^{n} (a_{ij}^{(c)} - dx_i y_j)^2$$

$$= \sum_{i,j=1}^{n} (a_{ij}^{(c)})^2 - 2d \sum_{i=1}^{n} s_i + d^2 \sum_{i=1}^{n} |x_i| \sum_{j=1}^{n} |y_j|$$

where we have defined

$$s_i = x_i \sum_{j=1}^{n} a_{ij} y_j.$$

Now, the first term in our expression for $r(x, y, d)$ is constant, and the signs of the x_i's do not affect the last term. Thus, to minimize r, the signs should be chosen so that all the s_i's have the same sign, positive if $d > 0$.

Now let J be the number of nonzero x_i's. Then

$$r(x, y, d) = \sum_{i,j=1}^{n} (a_{ij}^{(c)})^2 - 2d \sum_{i=1}^{n} s_i + d^2 J \sum_{j=1}^{n} |y_j|.$$

For a particular choice of x, the d which minimizes this is found by setting $\partial r(x, y, d)/\partial d = 0$. This gives

$$d = \sum_{i=1}^{n} s_i \Big/ \left(J \sum_{j=1}^{n} |y_j| \right).$$

Using this value gives

$$r(x, y, d) = \sum_{i,j=1}^{n} (a_{ij}^{(c)})^2 - d^2 J \sum_{j=1}^{n} |y_j|.$$

Therefore, minimizing $r(x, y, d)$ is equivalent to maximizing $d^2 J$. For a given number of nonzero x_i's, d is maximized by choosing those x_i's to be the ones corresponding to the largest sums $|\sum_{j=1}^{n} a_{ij} y_j|$. Thus, we have n possible choices of x to check: set the x_i's corresponding to the J largest $|\sum_{j=1}^{n} a_{ij} y_j|$ to $+1$ or -1 (sign chosen so that all s_i are positive), find d from the formula above, and choose the largest $d^2 J$ among the n choices for J.

ACKNOWLEDGMENT

We are grateful to A. Rosenfeld for helpful discussions and to W. K. Pratt for comments on a draft of the manuscript.

REFERENCES

[1] H. C. Andrews and C. L. Patterson, "Singular value decomposition and digital image processing," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, pp. 26–53, 1976.

[2] ——, "Outer product expansions and their uses in digital image processing," *Amer. Math. Mon.*, vol. 82, pp. 1–13, 1975.

[3] ——, "Outer product expansions and their uses in digital image processing," *IEEE Trans. Comput.*, vol. C-25, pp. 140–148, 1976.

[4] A. K. Jain, "Image data compression: A review," *Proc. IEEE*, vol. 69, pp. 349–389, 1981.

[5] C. B. Moler and G. W. Stewart, "An efficient matrix factorization for digital image processing," Dep. Comput. Sci., Univ. Maryland, College Park, 1978.

[6] W. K. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," *Proc. IEEE*, vol. 57, pp. 58–68, 1969.

[7] J. O. Thomas, "Digital imagery processing with special reference to data compression in remote sensing," in *Issues in Digital Image Processing* (Proc. 1978 NATO Adv. Study Inst. on Digital Image Processing), R. M. Haralick and J. C. Simon, Eds. Germantown, MD: Sijthoff & Noordhoff, 1980, pp. 247–290.