THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 57, No. 8, October 1978
Printed in U.S.A.

Some Extensions of the Ordering Techniques for Compression of Two-Level Facsimile Pictures

By F. W. MOUNTS, A. N. NETRAVALI, and K. A. WALSH

(Manuscript received September 2, 1977)

We present extensions of our earlier published ordering techniques for efficient coding of two-level (black and white) facsimile pictures. Ordering techniques use the two-dimensional correlation present in spatially close picture elements to change the relative order of transmission of elements in a scan line so as to increase the average length of the runs of consecutive black or white elements in the ordered line, making the data more amenable to one-dimensional run-length coding. The extensions that we consider allow us to use different run-length codes to match the statistics of different parts of the ordered data, and to drop certain runs from transmission. Computer simulations using the eight standard CCITT *pictures, which have a resolution of approximately 200 dots/inch, indicate that these extensions can result in transmission bit rates which are about 11 to 21 percent lower than the ordering schemes described in our earlier work. The entropies vary between 0.021 and 0.125 bits/pel for the eight pictures.*

I. INTRODUCTION

Coding of two-tone (black and white) facsimile pictures has gained considerable importance in the past few years, as is evidenced by a large number of papers as well as by a variety of facsimile communication systems. More and more sophisticated coding algorithms are being used which depend upon the two-dimensional spatial correlation present in picture data. This trend is understandable when one realizes that the cost of digital circuits and memories is decreasing faster than the cost of transmission.

This paper presents some extensions of our ordering schemes[1,2] for efficient coding of facsimile pictures. In the basic ordering scheme we make a prediction of the present element using the surrounding previously transmitted picture elements and classify it as "good" or "bad," depending upon the probability of the prediction being in error, conditioned on the specific values of the surrounding elements. We then change the relative order of the prediction errors corresponding to picture elements along a scan line using the "goodness" of the prediction in such a way as to increase the average run-length of the black and/or white elements and then transmit the run-lengths.

This paper has several objectives. First, we give the entropy results using our earlier ordering schemes on the CCITT (International Telegraph and Telephone Consultative Committee) images. This will allow a comparison with the many coding algorithms proposed by other workers since the CCITT images are widely available. This was not possible from the results presented in our earlier paper where we had used locally generated picture material. The second objective is to present certain extensions of the ordering schemes and give results of computer simulations. The following extensions are presented: (*i*) Since good and bad regions of the ordered prediction errors have different statistics, two sets of run-length codes can be used. It is not necessary to specify the location of the boundary between the good and bad regions to the receiver. (*ii*) Runs across the good-bad region boundary can be bridged wherever advantageous, even if the color of the element changes across the boundary. (*iii*) A specified run in each line of data can be omitted from transmission since the number of elements in a line is fixed. The length of the omitted run can be derived at the receiver if a line sync code is transmitted at the end of each line.

Computer simulations indicate that entropies ranging between 0.021 and 0.125 bits/pel for the eight CCITT pictures are possible using these extensions. This represents a 11- to 21-percent decrease over the ordering techniques of our earlier paper.[1]

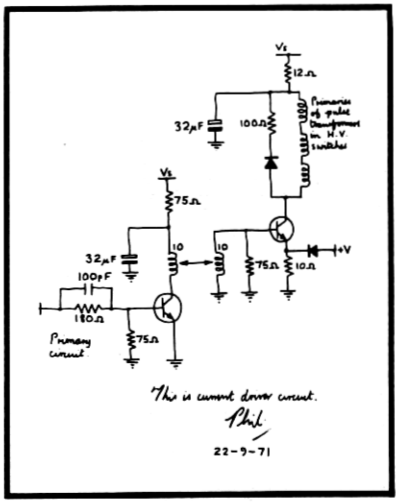
II. CODING ALGORITHMS

In this section, we describe our coding algorithms in detail and present results of the computer simulations. The pictures used for simulations are the eight CCITT pictures which have a resolution of approximately 200 dots/inch. Each picture consists of 2128 lines with 1728 picture elements (pels) in each line. Copies of these pictures are shown in Figs. 1a through 1h. As a measure of performance, we used the sample first-order entropy of run-length statistics. We computed the average black and white run-lengths and the entropy of black and white runs using, for example, the formula

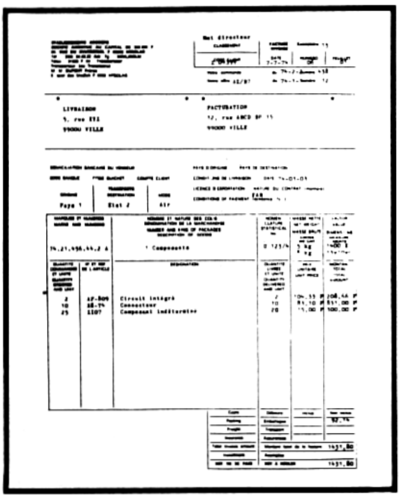
$$E_w = -\sum_i \frac{n_i}{N} \cdot \log_2 \frac{n_i}{N}, \tag{1}$$

where E_w is the entropy of the white run-lengths, n_i is the number of white runs of length i, and N is the total number of white runs. Using these and eq. (2), we computed the entropy, E, in bits/pel by:

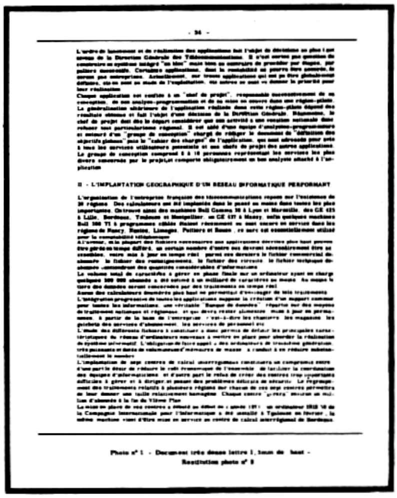
(a)



(b)



(c)

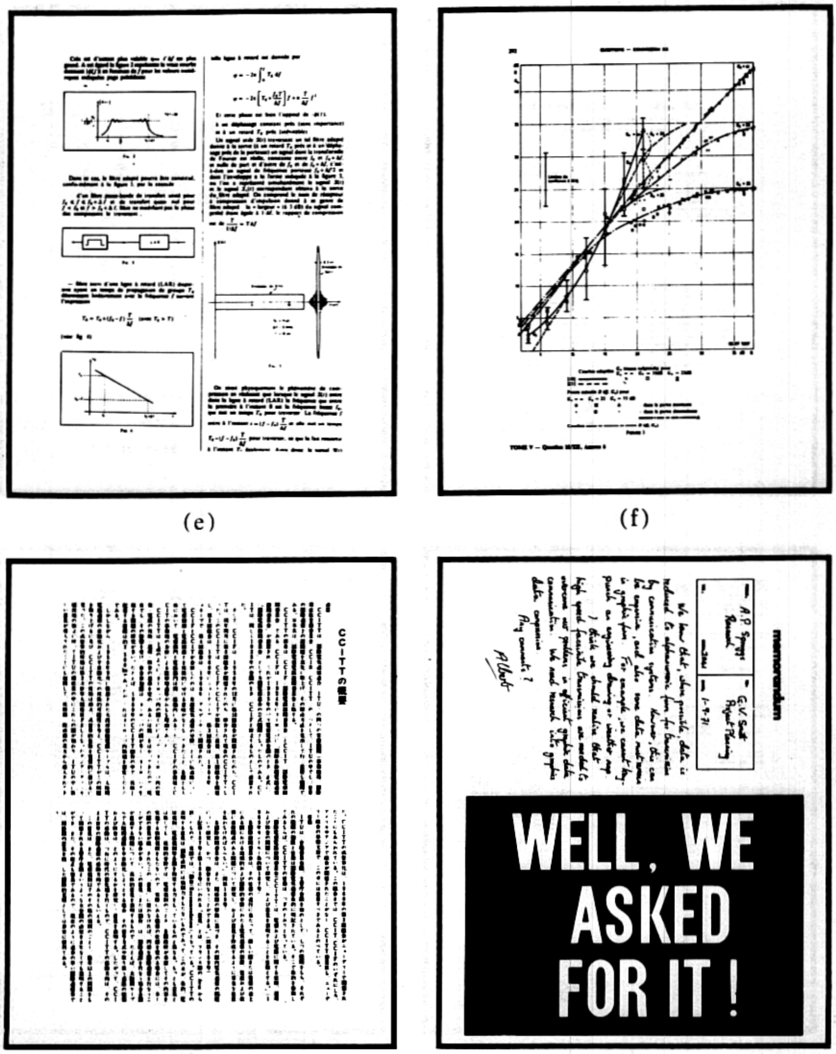


(d)

Fig. 1—The eight (a to h) CCITT pictures used for computer simulation. Each picture consists of 2128 lines with 1728 pictures elements in each line and has an approximate resolution of 200 dots/inch. (*Figs. 1e through 1h on next page*)

$$E = \frac{E_w \cdot N_w + E_b \cdot N_b}{r_w N_w + r_b N_b}, \tag{2}$$

where E_b is the entropy of the black run statistics, r_w, r_b are the average white and black run-lengths, respectively, N_w, N_b are the number of white and black runs, respectively, and E is the entropy in bits/pel. The above numbers are computed for the entire picture (1728 × 2128 pels) using, on the sides and top of the picture, a border of white elements surrounding the actual picture.



(e) (f) (g) (h)

Fig. 1 (*Continued from previous page*).

2.1 Prediction algorithm

The first step in the ordering algorithm consists of making a prediction of the present picture element using the already transmitted surrounding picture elements. We define a state S_i using the four surrounding picture elements $\{X_j\}_{j=1,\ldots,4}$ as shown in Fig. 2. There are 16 states. The predictor is developed in a standard way[3–5] as the one which minimizes the probability of making an error, given that a particular state has occurred. Thus the predictor $C(S_i)$, for a given state S_i, is given by:

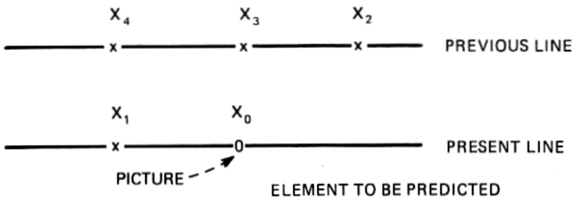


Fig. 2—Configuration for state definition.

$$C(S_i) = \text{“black,” if } P(X_0 = \text{“black”} \,|S = S_i) > 0.5$$
$$= \text{“white,” otherwise,}$$

where $P(\cdot|\cdot)$ is the conditional probability measured for the picture. For convenience, we represent the color of the picture elements by "1" and "0," "1" for black and "0" for white. The predictor varies from picture to picture; however, the variation is not great, as shown in our earlier paper.[1] The predictor for a typical picture [CCITT picture 2 (Fig. 1b)] is shown in Table I.

2.2 Ordering algorithms with one set of run-length codes

In this section, we give the simulation results using our earlier ordering algorithms. First, in Table II, for the purposes of comparison, we give the entropies of the run-length statistics from the raw picture data as well as from the prediction error data. As expected, the entropies of the run-lengths of the prediction errors show about 0.7 to 24 percent decrease over the entropies of the run-lengths of raw data. The decrease is smaller for the busier pictures such as the CCITT pictures 4 and 7.

Next, we simulated the ordering algorithm of Ref. 1. As explained there, this algorithm can be illustrated by considering a memory containing 1728 cells (equal to the number of elements per line). Let the cells

Table I—State-dependent prediction for CCITT picture 2 (Fig. 1b)

	State S_i				$P(X_0/S_i)$		Predicted
	X_1	X_2	X_3	X_4	$X_0 = 0$	$X_0 = 1$	Value X_0
S_0	0	0	0	0	1.000	0.000	0
S_1	1	0	0	0	0.300	0.700	1
S_2	0	1	0	0	0.777	0.223	0
S_3	1	1	0	0	0.006	0.994	1
S_4	0	0	1	0	0.822	0.178	0
S_5	1	0	1	0	0.055	0.945	1
S_6	0	1	1	0	0.323	0.677	1
S_7	1	1	1	0	0.001	0.999	1
S_8	0	0	0	1	1.000	0.000	0
S_9	1	0	0	1	0.690	0.310	0
S_{10}	0	1	0	1	0.971	0.029	0
S_{11}	1	1	0	1	0.154	0.846	1
S_{12}	0	0	1	1	0.996	0.004	0
S_{13}	1	0	1	1	0.200	0.800	1
S_{14}	0	1	1	1	0.708	0.292	0
S_{15}	1	1	1	1	0.012	0.988	1

Table II—Entropy comparisons for different coding algorithms. The entropy numbers do not include certain housekeeping bits (e.g., line sync, color of the beginning run in a line)

Coding Algorithm		Entropy (bits/pel) CCITT Image Number							
No.		1	2	3	4	5	6	7	8
1	One-dimensional run-length coding	0.0505	0.0447	0.0914	0.1652	0.0988	0.0679	0.1791	0.0870
2	Run-length coding of prediction errors	0.0466	0.0373	0.0693	0.1640	0.0795	0.0482	0.1678	0.0678
3	Run-length coding of ordered prediction errors; one set of codes, goodness threshold = 0.1.	0.0390	0.0267	0.0571	0.1396	0.0652	0.0366	0.1400	0.0463
	Run-length coding of ordered prediction errors; all "white" state = good state	0.0398	0.0305	0.0592	0.1424	0.0673	0.0392	0.1442	0.0569
4	Run-length coding of ordered prediction errors; two sets of codes, goodness threshold = 0.1, good—bad boundary run broken	0.0356	0.0247	0.0547	0.1287	0.0613	0.0351	0.1284	0.0430
5	Run-length coding of ordered prediction errors; two sets of codes, goodness threshold = 0.1, good—bad boundary bridged	0.0351	0.0233	0.0527	0.1282	0.0596	0.0335	0.1274	0.0419
6	Run-length coding of ordered prediction errors; one set of codes, goodness threshold = 0.1, first run dropped	0.0338	0.0201	0.0501	0.1320	0.0579	0.0298	0.1326	0.0390
7	Run-length coding of ordered prediction errors; two sets of codes, goodness threshold = 0.1, good—bad boundary bridged; last decodable run dropped	0.0324	0.0210	0.0506	0.1239	0.0569	0.0312	0.1250	0.0398

of this memory be numbered from 1 to 1728. We classify the states used for predictors into two categories, good or bad. Good states are those for which the probability of the prediction being in error, conditioned on that state, is less than a given threshold (defined as the goodness threshold). All the other states are bad. In the process of ordering, if the first element of the present line has a state which is classified as good, we put the prediction error corresponding to it in memory cell 1; if, on the other hand, the state is classified as bad, we put the prediction error in memory cell 1728. We continue in this manner: the prediction error for the ith element of the present line is put in the unfilled memory cell of the smallest or the largest index, depending on whether the state corresponding to the ith element is good or bad. When the memory is filled, its cells are read in numerical order and the contents are run-length encoded. It is easy to see that the present line can be uniquely reconstructed from the knowledge of the run-lengths of the ordered line, since the ordering information is known to the receiver. The efficiency of such ordering depends upon the threshold used for classifying the states into good or bad. Table II shows two examples, one in which the goodness threshold was 0.1 and the other in which only one state (corresponding to all four surrounding elements being zero) is classified as good. A goodness threshold of 0.1 appears to be acceptable among the many thresholds that we used in our simulations. Comparing entropies corresponding to the ordered and unordered prediction errors, we see that ordering reduces the entropy by about 15 to 32 percent, depending on the picture used. Also, ordering of the prediction errors brings entropies down by 15 to 47 percent of the run-length coding of raw data. It should be noted that in each of the above cases the predictor was optimized for the particular picture.

2.3 Ordering algorithms with two sets of codes

Statistics of the run-lengths in the good and bad regions of the ordered prediction errors are quite different. As an example, for CCITT picture 2 (Fig. 1b), 98.5 percent of the pels fall in the good region of which 99.9 percent are correctly predictable, whereas the bad region contains only 1.5 percent of the total elements of which 73 percent are correctly predictable. Thus, the average run-lengths in the good region are much larger than in the bad region. Such a variation in the statistics can be exploited by using two different sets of run-length codes for the good and bad regions, respectively. The algorithm* would then operate as follows: First, we put the ordered prediction errors in the memory as before; then, the contents of the memory are run-length coded with one set of codes in the good region and a different set of codes in the bad region.

* This algorithm is related to the one proposed by Preuβ (Ref. 5). It is discussed here mainly for completeness and was motivated by the communication we received from him (Ref. 6).

Switching from one set of codes to the other is done at the boundary of the good-bad region even though the ordered line may not have a new run at the boundary. This process will break the run at the boundary between the good and bad region of the ordered line, whereas the ordering technique discussed in Section 2.2 continues the run (whenever possible) across the boundary of the good-bad region. This procedure is continued until all the runs from the memory are exhausted.

At the receiver, the coded run-lengths for a complete line are held in a memory. Good or bad runs are decoded from the memory as needed.

The results of computer simulations for the ordering scheme with two sets of codes are shown in Table II. These results use a goodness threshold of 0.1. Comparing the entropies from algorithms with one and two sets of codes, it is seen that with two sets of codes about 4 to 8 percent improvement is possible. This is the opposite conclusion* from that given in our earlier paper, which used a different source material.[1] For the pictures used in Ref. 1, we had found that ordering schemes with two sets of codes resulted in 10 to 18 percent higher entropies than the entropies obtainable with one set of codes. This may have been a result of the small size of the pictures used for the simulation (an array of 256×256 picture elements).

2.4 Ordering algorithms with two sets of codes and bridging of good-bad boundary

Use of two sets of run-length codes described in the previous subsection resulted in the breaking of a run at the boundary of the good-bad region since part of the run may be in the good region and the other part may be in the bad region. To avoid breaking the run, which extends across the boundary, we code the boundary run using the run-length code of the good region or the bad region as follows: If the boundary run is first required as a bad run in the process of decoding the run-lengths at the receiver, it is coded as a bad-run; otherwise, it is coded as a good run. The method in which the receiver decodes the bridged run is similar to the one given in the next subsection. Results of such a scheme are shown in Table II. Bridging of the run across the boundary results in an improvement of about 0.39 to 6 percent over nonbridging. As would be expected, the percent improvement is smaller for busier pictures.

2.5 Ordering algorithms with dropped runs

In most facsimile communication systems a code for the line sync is sent at the end of each line of coded data. Since the number of elements in a line is fixed, this is redundant. A run can be dropped from each line

* We thank D. Preuß for showing us data from his simulations which first demonstrated this fact.

as long as the receiver knows the position of the dropped run. In the ordered line a large benefit can be derived by dropping the first good run, since it is generally the longest. This also avoids transmission of run-length codes for lines with no prediction errors. Table II shows results of the simulation of a scheme in which the first good run from the ordered prediction errors is dropped from transmission, and the rest of the runs are transmitted by using one set of run-length codes. Dropping the run reduces the entropy to between 0.020 and 0.133 bits/pel which is a 5 to 25 percent reduction compared to the case where all the runs are sent.

It is also possible to drop a run from transmission when two sets of codes are used for the run-lengths in the good or bad regions. In this case, the first run cannot be dropped since the receiver switches between the two sets of codes depending on the past decoded data. However, the last run that the receiver needs to decode may be dropped. We have simulated a scheme in which the good-bad region boundary is bridged and the last decodable run is dropped. To explain the scheme, consider a line made up of run-lengths of ordered prediction errors as shown in Fig. 3. We use two sets of codes and start transmitting codewords corresponding to run-lengths G_1, G_2, $\cdots$, B_2, B_1 of the ordered line, appropriately switching the code in the good and bad regions. The receiver decodes these run-lengths as needed. To bridge the boundary run and drop the last decodable run, we use the following rules:

(*i*) If there are no runs in the good region, drop the last run in the bad region, i.e., B_m.

(*ii*) If there are no runs in the bad region, drop the last run in the good region, i.e., G_n.

(*iii*) If the last two runs required by the receiver in the decoding process are G_n and B_m (in either order), drop the runs G_n and B_m. This is done independently of the color of prediction errors in G_n and B_n.

(*iv*) If the last two runs required by the receiver are from the bad region and at least one good region run has occurred, then if

- (*a*) color of B_m is a "1," bridge G_n and B_m, code it using the good region code, and drop B_{m-1}.
- (*b*) color of B_m is a "0," drop B_m.

(*v*) If the last two runs required by the receiver are from the good region and at least one bad region run has occurred, then if

- (*a*) color of G_n is a "1," bridge G_n and B_m, code it using a bad region code, and drop G_{n-1}.
- (*b*) color of G_n is a "0," drop G_n.

Rules (*iv*) and (*v*) allow us to drop a run of 0s rather than a run of 1s, since runs of 0s usually have longer lengths than runs of 1s. Also, it is possible to bridge the runs at the boundary independent of the color change across the boundary of the good and bad region. Thus, the above

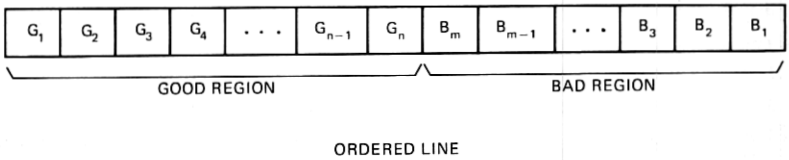


Fig. 3—Ordered run-lengths.

strategy allows dropping a run from transmission, bridging runs across the boundary (whenever it is advantageous, even if colors change), and the use of two separate sets of codes for the good and bad regions.

At the receiver, the coded run-lengths are held in memory and decoded as needed. A running total of the number of elements from decoded run-lengths is kept. If all the run-lengths have been decoded from the receiver memory and an additional run is required, this running total is subtracted from the total number of elements in a line, and the result is taken as the length of the next run. If the result is zero, then the next run is taken to be of opposite color, as usual, and decoding proceeds until the end of the line. The simulations using the above scheme decreased the entropy to between 0.021 and 0.125 bits/pel as shown in Table II. For busy images this scheme does better than the scheme which uses only one set of codes and drops the first run. However, for quieter pictures the performance is reversed.

III. DISCUSSION AND SUMMARY

We have described in this paper schemes for efficient coding of two-level (black and white) facsimile pictures. These were extensions of our earlier schemes which ordered the prediction errors before run-length coding. The most sophisticated extension presented here results in an entropy of between 0.021 and 0.125 bits/pel. Our computer simulations indicate that use of two sets of codes for good and bad regions of the ordered pictures results in about 4 to 8 percent decrease in entropy compared to using only one set of codes; whereas using two sets of codes, bridging the good-bad boundary run, and dropping the last decodable run decreases the entropy by 11 to 21 percent.

It should be mentioned that this is not a definitive coding system study. We have not considered many important factors crucial to the success of any coding system such as the run-length codes and their picture dependence and the effect of transmission errors.

IV. ACKNOWLEDGMENTS

We would like to thank Dr. Ronald Arps of IBM—Los Gatos, California, and Professor Hans G. Musmann of Technical University of Hanover, West Germany, for supplying us with the digitized CCITT images. Special thanks are due to Dr. Dieter Preuβ, also of Technical

University of Hanover, West Germany, who sent us several communications describing results of his simulations which helped clear the differences between our earlier results[1] and those given in Section 2.3.

REFERENCES

1. A. N. Netravali, F. W. Mounts, and E. G. Bowen, "Ordering Techniques for Coding of Two-Tone Facsimile Pictures," B.S.T.J., *55,* No. 10 (December 1976), pp. 1539–1552.
2. A. N. Netravali, F. W. Mounts, and J. D. Beyer, "Techniques for Coding Dithered Two-Level Pictures," B.S.T.J., *56,* No. 5 (May–June 1977), pp. 809–819.
3. J. S. Wholey, "The Coding of Pictorial Data," IRE Trans. Information Theory, *1T-7* (April 1961), pp. 99–104.
4. H. Kobayashi and L. R. Bahl, "Image Data Compression by Predictive Coding," IBM J. Res. Devel., 1974, pp. 164–179.
5. D. Preuβ, "Comparison of Two-Dimensional Facsimile Coding Schemes," Int. Conf. Commun. (June 1975), pp. 7-12–7-16.
6. D. Preuβ, private communication.