CR 86158

NAS12-554

INTERIM SCIENTIFIC REPORT

DEVELOPMENT OF ADVANCED DIGITAL TECHNIQUES FOR DATA ACQUISITION PROCESSING AND COMMUNICATION

By S. M. Sussman, E. P. Greene, R. Zaorski

15 March 1969

Prepared under Contract No. NAS12-554

National Aeronautics and Space Administration
Electronics Research Center
RC/Computer Research Laboratory
Cambridge, Massachusetts

ADVANCED COMMUNICATIONS INFORMATION MANAGEMENT

RESEARCH • DEVELOPMENT • ENGINEERING

ADCOM

A TELEDYNE COMPANY

Dr. Edwin H. Farr
Technical Monitor
NAS12-554
575 Technology Square
Cambridge, Massachusetts 02139

Requests for copies of this report should be referred to:

National Aeronautics and Space Administration
Electronics Research Center
RC/Computer Research Laboratory
Cambridge, Massachusetts 02139

NAS12-554

INTERIM SCIENTIFIC REPORT FOR THE DEVELOPMENT OF ADVANCED DIGITAL TECHNIQUES FOR DATA ACQUISITION PROCESSING AND COMMUNICATION

By S. M. Sussman
E. P. Greene
R. Zaorski

15 March 1969

Prepared under Contract No. NAS12-554 by

ADCOM
A Teledyne Company
808 Memorial Drive
Cambridge, Massachusetts 02139

For

National Aeronautics and Space Administration
Electronics Research Center
RC/Computer Research Laboratory
Cambridge, Massachusetts 02139

G-98IS

PRECEDING PAGE BLANK NOT FILMED.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

LIST OF TABLES

INTERIM SCIENTIFIC REPORT FOR THE DEVELOPMENT OF ADVANCED DIGITAL TECHNIQUES FOR DATA ACQUISITION PROCESSING AND COMMUNICATION

By S. M. Sussman, E. P. Greene and R. Zaorski

ADCOM
A Teledyne Company
Cambridge, Massachusetts 02139

SUMMARY

This interim scientific report covers work performed in the following areas:

1) The development of fidelity criteria for automatic evaluation of image quality.
2) The analysis of various methods for encoding the significant sample point timing information in one-dimensional data compression algorithms.
3) The development of two-dimensional data compression algorithms.

Two approaches to Fidelity Criteria are described in terms of motivation for their selection and computer program implementation. One approach is based on the statistics of areas consisting of contiguous picture elements of the same intensity. The second approach derives the statistics of error-runlengths along a scan line.

The material on Time Sequence Coding is devoted to a discussion of various methods for conveying the time information that specifies the location of significant samples along a scan line. The methods are compared with respect to efficiency; i. e., maximization of bit compression ratio for different levels of image redundancy. The problem of error control for the timing data is treated. Simulation results of the error performance of a particular code on burst error channels are presented.

Two-Dimensional Data Compression Algorithms are described for taking advantage of image correlation in both horizontal and the vertical direction. Two approaches are discussed in which the picuture is divided into squares comprising a small number of picture elements. In one case the image is represented by a set of four "edge" patterns on 4 x 4 square and a form of zero-order prediction is employed for square-to-square correlation. In the other case a finite set of normalized intensity patterns is employed provided the correlation with the actual pattern over the square is sufficiently high,

INTRODUCTION

This interim scientific report covers work performed under Contract NAS 12-554 (DOC9) Development of Advanced Digital Techniques for Data Acquisition Processing and Communication. Activity during the reporting period was concentrated in the following areas:

1) The development of fidelity criteria for automatic evaluation of image quality.

2) The analysis of various methods for encoding the significant sample point timing information in one-dimensional data compression algorithms.

3) The development of two-dimensional data compression algorithms.

The next section discusses the fidelity criteria that have been under investigation. Two approaches are described in terms of motivation for their selection and computer program implementation. One approach is based on the statistics of areas consisting of contiguous picture elements of the same intensity. The area statistics of an image before and after perturbation will be examined as a potential fidelity criterion. The second approach derives the statistics of error-runlengths along a scan line. An error run is defined as a sequence of consecutive points whose difference between original and perturbed version exceeds a prescribed tolerance.

The section on Time Sequence Coding is devoted to a discussion of various methods for conveying the time information that specifies the location of significant samples along a scan line. The methods are compared with respect to efficiency; i.e., maximization of bit compression ratio for different levels of image redundancy. The problem of error control for the timing data is treated. Simulation results of the error performance of a particular code on burst error channels are presented.

The final section on Two-Dimensional Data Compression Algorithms describes two techniques for taking advantage of image correlation in both horizontal and the vertical direction. In both cases the picture is divided into squares comprising a small number of picture elements. In one case the image is represented by a set of four "edge" patterns on a 4 x 4 square and a form of zero-order prediction is employed for square-to-square correlation. In the other case a finite set of normalized intensity patterns is employed provided the correlation with the actual pattern over the square is sufficiently high. If so, the pattern identity is transmitted, if not, the actual element values are sent. The second case assures that the mean-square approximation error over the square is held within a specified bound.

FIDELITY CRITERIA

Problem Statement

In most instances the final destination of image data is a human observer. This has led to the virtually exclusive use of visual examination as the mechanism for establishing the fidelity of images which have undergone perturbation through transmission or other forms of processing. Subjective evaluation of images can be a time-consuming process and is susceptible to the variabilities of human observers. In spite of the drawbacks of subjective evaluation, no acceptable substitute is currently available.

The aim of this portion of the study effort is to conceive and test methods for automatic evaluation of image fidelity. To be useful these methods should 1) provide ratings or fidelity criteria which can be calibrated in terms of human observers for a particular class of images, and 2) yield evaluations consistent with human observers for other images from the same class.

The approach employed in this study is to base the automatic evaluation on two digitized versions (e. g., magnetic tape) of an image - an original and a perturbed or degraded replica. The evaluation processor software operates upon each image individually or upon their point-by-point difference to arrive at a fidelity measure. Two methods for generating fidelity criteria have been investigated in this study and the necessary computer programs have been written. The next two subsections present the background and motivation for each scheme.

Following that are descriptions of the two measurement programs in the form of FORTRAN subroutines AREA and XTLR. In general, the computer accepts raw data from a scanner/plotter magnetic tape under control of a main program which calls one of the measurement subroutines to operate on the data. The reduced data appears as a tabulation which is suitable for further analysis by mathematical methods.

The measurement programs have been debugged and tested with dummy data. Program listings of subroutines AREA and XTLR are presented in Appendix A. The analysis step in the evaluation of picture fidelity now awaits receipt of actual output tapes from the scanner/plotter.

Area Statistics

Experiments have shown that human visual perception is primarily based on areas and boundaries in images. It is appropriate therefore to examine area criteria as opposed to point criteria to derive a measure of image fidelity. In particular, the area statistics of images, before and after perturbation, are being investigated.

We define an area as a collection of contiguous image points having the same intensity within a preset tolerance. The area size is the number of image points contained in a single area. The simplest area statistic is the first order frequency distribution of area size without regard to intensity differences among areas of the same size. This distribution is clearly sensitive to image degradation as evidenced by the following two examples.

The addition of random noise to an image is visible as "snow" and has the effect of breaking up large areas while creating new small areas. This is reflected in the area statistics as an increase in the total number of areas and a shift in the distribution toward smaller areas.

A second illustrative source of image degradation is a reduction in the intensity resolution of a digitized picture. When the number of quantum levels of intensity becomes sufficiently small the effect known as contouring arises. Contouring refers to the appearance of artificial sharp boundaries in regions of an image where a gradual transition of intensity would be evident in the original picture. The reduced resolution forces the transitions to occur in pronounced steps with constant intensity in between. The recurrence of the steps on successive scan lines produces noticeable contour boundaries. This form of image degradation causes a decrease in the total number of image areas and a greater proportion of larger areas. Thus both random noise and reduced resolution are reflected in definitive changes in the first-order area statistics of an image.

In a somewhat different context (evaluation of image information content) the area statistics of a cloud cover photograph taken from an orbiting spacecraft were computed by Massa and Ricupero.[1] The normalized cumulative distribution is plotted in Fig. 1 for resolutions ranging from 1 to 6 bits (2 to 64 levels). The total area number (A_{tot}) and other parameters of the normalized curves could serve as

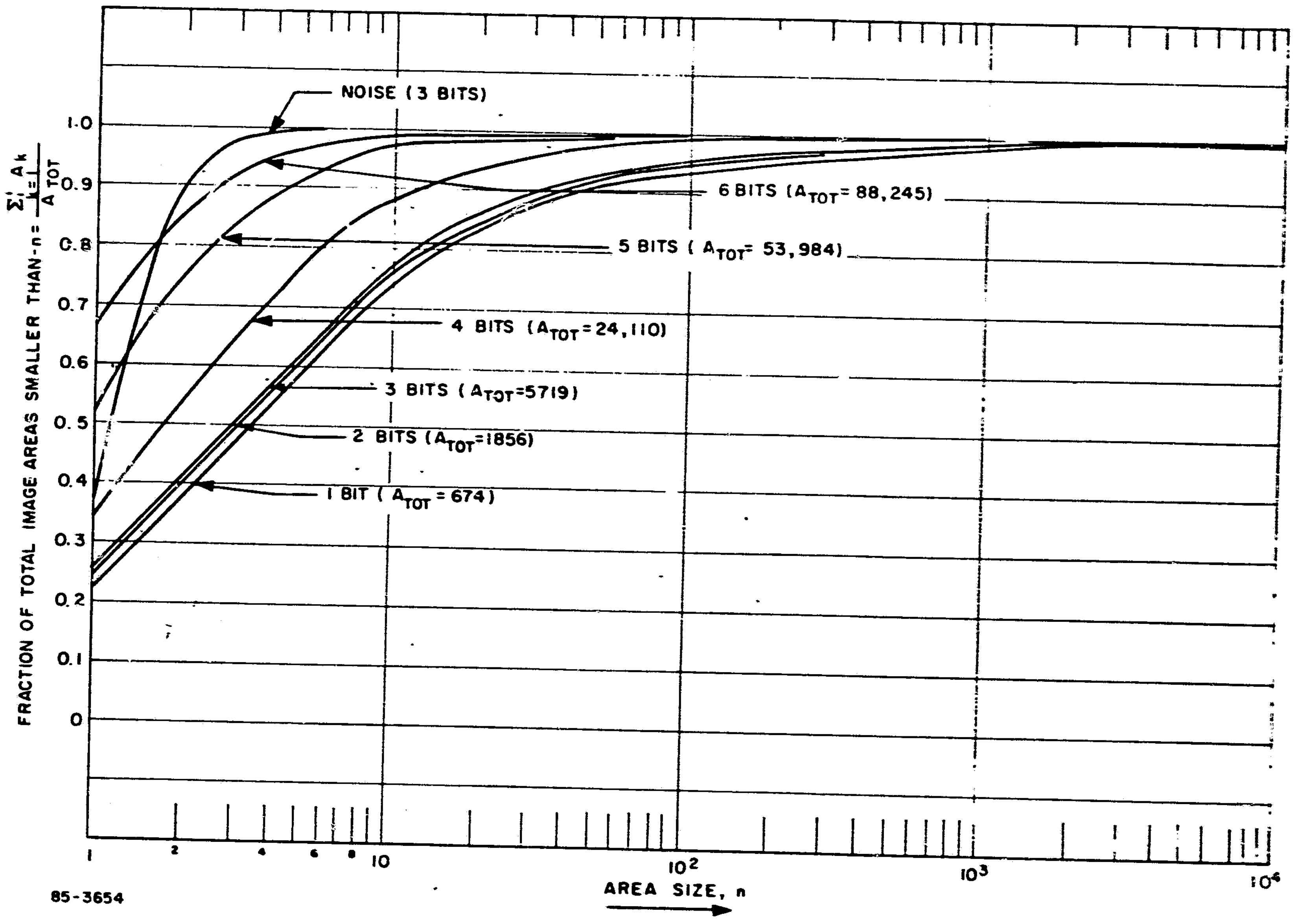


Fig. 1 "Cloud" Area Properties as a Function of Quantization Level

Error-Run Statistics

Subjective assessment of images is significantly dependent on the degree of spatial correlation of image noise or errors as well as on the amplitude of the noise. For a fixed ratio of signal to rms noise, the acceptability of an image has been found to depend on the two-dimensional spatial correlation function (or bandwidth) of the noise, relative to the corresponding local correlation for the image itself. We emphasize "local" since the effect of the noise varies over different regions of a given picture depending upon the characteristics of the image regions.

Preliminary consideration has been given to the exploitation of these phenomena in devising fidelity criteria. Ideally, one would like to compute the autocorrelation of both the image and the additive noise (difference between original and perturbed version). However, both processes are in general non-stationary and one would have to compute the local correlations for subdivided regions of a picture. The computational burden for executing this procedure appears to be prohibitive.

As an alternative and considerably simpler approach, one that still retains the feature of spatial correlation -at least in the direction of scan - attention has been focussed on error runlengths. An error-run consists of a sequence of consecutive points on a scan line for which the difference between the original and perturbed image exceeds a specified amplitude tolerance. The basic amplitude criterion is modified to allow for possible displacement of amplitude transitions along a scan line by an amount small enough to be not objectionable. The processing operations consist of a line -by-line comparison of original and perturbed image and the derivation of error-run statistics. Each point on the perturbed image is compared with its corresponding element on the original and with those elements displaced a few points (1 or 2) on either side. If a match within amplitude tolerance fails to occur for any of these elements an error-run is initiated. Each subsequent point is treated similarly and consecutive points in error are counted. The frequency distribution of runlengths is tabulated for analysis as a fidelity criterion. Although this criterion ignores correlation of errors from line to line, it is especially appropriate for evaluation of one-dimensional data compression algorithms where errors tend to appear as streaks along the direction of scan.

Description of Subroutine AREA

Introduction. - The program called AREA is a FORTRAN subroutine designed to input raw data in the form of a digital representation of video scan lines and to reduce this data to a form amenable to mathematical manipulation. In particular, contiguous picture areas of the same tonal value (or brightness) are measured by size and counted. For a given frame (picture), the data output from subroutine AREA is a two dimensional matrix whose elements yield the number of contiguous areas encountered for each size and tonal value. The following sections describe the method employed and provide information on the use of this subroutine.

Method. - Each time that subroutine AREA is called, it inputs a vector corresponding to one scan line of the video frame. The raw data is reduced to output form on a line-by-line basis. Using the current scan line as a frame of reference, the various areas (of differing gray tones) making up the total frame will be seen as newly emerging, continuing, or disappearing line segments in the scan line. Except for the first and last scan lines, all three conditions exist at most times, and it is the function of this program to keep track of the changes and to provide a summary of size, number, and tonal value parameters.

The fundamental operation of this subroutine is the comparison of each "new" vector against the previous "old" vector in order to determine the continuity or discontinuity of picture areas. A line segment in the new vector is defined as those contiguous points having the same tonal value. As a new vector is read in, it is assembled into working matrix MA segment-by-segment. Both the tonal value of the line segment and the position of its end point are stored.

Continuity is determined by comparing, sequentially, the line segment tonal value and end point position in MA with similar parameters in working matrix MB which corresponds to the "old" vector. Both MA and MB contain storage of a third parameter, the area locater, which for MA is initially zero for every iteration (new scan line). The area locater is an index number which relates each segment in MB with a counter in working vector MD containing the accumulated area to which an MB segment belongs. If a segment in MA is continuous on a segment in MB, it is given the same locater and the program proceeds to the next pair of line segments.

There are several complicated conditions that might exist while comparing an MA segment with an MB segment, but all are reduced (by various switches and operations) to three simple cases. Besides continuity, the program determines whether the current segment in MA is newly emerging, or whether the current MB segment is disappearing, or both. In either case, working matrix MC comes into play as the table that controls the assignment of locations available in MD for accumulating areas. Entries in the data output matrix are made when areas disappear.

Usage. - The matrices and vectors employed by this subroutine are typed as integers for specific design reasons, and the types of the array names in the calling program must agree in this requirement.

Since the size of the largest frame of raw data is expected to be 512 by 512 and since the area size is one dimension of the output matrix, the numerical value of the measured area size is reduced by an arithmetic operation in order to minimize storage requirements. The area sizes are divided into the categories of 1 to 100, 101 to 10000, and over 10000. In the first category, there is a one to one correspondence between the area size and the output matrix index number. In the second category, a geometric progression generates "bins" to which area sizes are assigned: Sizes greater than 10000 are stored, along with their tonal values, in a separate matrix.

The following table identifies each of the dummy arguments for this subroutine. Some of these arguments are further discussed, as required, in subsequent paragraphs.

CALL AREA (MN, MATRIX, N1, N2, LINE, N, MA, MB, MC, MD, M, LARGE, N3, Y)

where

MN	is the number of lines per frame
MATRIX	is the name of the output data matrix
N1	is the number of rows in MATRIX
N2	is the number of columns in MATRIX
LINE	is the name of the input vector
N	is the length of LINE
MA, MB	are working matrices of size M by 3

MC, MD	are working vectors of length M
M	is the working length as noted above
LARGE	is the name of the overflow storage
N3	is the length of LARGE
Y	is the bin size factor

The dimension, N1, of MATRIX depends on the expected sizes of areas and the bin size accuracy desired. If N1 is greater than 100, the relationship is: $Y = \exp\left((\log 100)/(N1-100)\right)$. For example, if N1 is 200, then Y = 1.047. If N1 is less than 100, then areas exceeding size 100 appear in the overflow to LARGE. The other index for MATRIX is derived from the input vector by adding one to the tonal value. Therefore, tonal values of zero are permitted, negative values are not, and MATRIX dimension N2 is given by the number of possible tonal values.

The user must also determine the value of M for dimensioning the working matrices. M must be greater than the largest number of line segments (or transitions in tone) in any one line of the frame.

Description of Subroutine XTLR

Introduction. - The program called XTLR is a FORTRAN subroutine designed to compare one scan line each from a transmitted picture and its original. This subroutine, when called repeatedly by the main program, tabulates discontinuites in the transmitted scan lines that do not occur in the original, or vice versa. Under visual observation, these errors may appear as "snow" or "streaks" that degrade the picture fidelity. While the subjective appearance might vary, these errors consist of segments of a scan line that have (1) an intensity error, or (2) a displacement (timing) error, or (3) both intensity and displacement errors.

Subroutine XTLR detects both types of errors and tabulates them according to the length of the line segment which is in error. The frequency distribution of error runlengths thus obtained serves as an objective measurement of picture fidelity.

Method. - The scan lines are treated as vectors by this subroutine, and two vectors are compared point-by-point. The displacement tolerance allowed results in the comparison of a point in the vector under test with the points in a line segment of the standard (original) vector. The length of this segment depends upon the tolerance specified by the user.

Tolerances are also allowed for the absolute magnitude of the difference between the value of the test point and the value of any point in the segment of the standard vector. Adjacent points in error are counted and stored in an output vector on the basis of the error length.

Usage. - The two input vectors and the one output vector must be dimensioned by the user. Note that these vectors are typed as integers. The dummy arguments of the calling statement are explained below:

```
CALL XTLR (MR, LA, LB, N, M, MAX)
```

where	MR	is the output vector
	LA	is the standard input vector
	LB	is the input vector to be tested
	N	is the length of all three vectors
	M	is the displacement tolerance
	MAX	is the value tolerance

ANALYSIS OF TIME SEQUENCE CODING

Introduction

The transmitted output from any data compression system contains at least two types of data:

1) Non-redundant sample data, and

2) Timing information associated with the non-redundant samples.

In addition to these two types of data, periodic synchronization patterns and possibly sensor identification tags would be required for typical applications. This chapter deals with an analysis of various possible coding methods designed to provide the time tagging information.

In typical video applications the number of uncompressed samples along a given scan line will usually be between 200 and 525. We will assume that a special code word is introduced into the message to signify the start of a given scan line. The time tagging process then consists of merely identifying the position along the scan line at which each non-redundant sample was obtained. In the following sections various means of accomplishing this objective will be analyzed and compared. As a basis for comparing the overall effectiveness of typical codes the following parameters will be used:

No. of data bits required per non-redundant sample = 5*
No. of sample positions per scan line = 500.

Define N_s to be the sample compression ratio for a given scan line. Let C_D be the total number of bits necessary to encode the data section of all non-redundant samples from the scan line and C_T be the total number of time tagging bits required within the scan line. Then the bandwidth compression ratio (N_{BW}) for the scan line is given by

$$N_{BW} = \frac{5(500)}{C_T + C_D} \qquad (1)$$

* This assumes that the number of data bits necessary per sample is the same for both the uncompressed and compressed data cases. In most cases this is valid but there are some algorithms (such as FOIOOT) which require an additional bit in the data field that would not be needed if the data were transmitted in uncompressed format. This effect will not be considered in this analysis.

But $$C_D = \frac{5(500)}{N_s} \qquad (2)$$

Hence $$N_{BW} = \frac{N_s}{1 + \frac{N_s C_T}{2500}} \qquad (3)$$

Single Bit per Sample Time Tag Coding

Probably the simplest conceptual approach to time sequence coding would be to encode the scan line as a 500 bit time tag field followed by a variable length section consisting of the non-redundant samples. Such a format is shown below.

Time Sequence Field (500 bits)	Non-Redundant Data Field				
1000010010..........01001	D_1	D_2	D_3	....	D_N

The occurrence of a "1" in a given position of the 500 bit time sequence field will be taken to indicate that a non-redundant sample was found in the corresponding position within the scan line. With the coding scheme the number of bits required for time tagging is constant and consists of one bit per sample. The bandwidth reduction ratio for this approach is

$$N_{BW} = \frac{5N_s}{5 + N_s} \qquad (4)$$

The major drawback of this scheme is that the coding for the time sequence field is very inefficient. Typically one expects that $N_s >> 2$. In such a case the 500 bit field will contain mostly "0" bits with only occasional "1" bits. This results in a low entropy per transmitted bit.

Optimum Time Sequence Coding

The coding scheme described above serves to illustrate a simple but inefficient coding scheme. We will now immediately jump to the opposite extreme and analyze an ideal coding scheme which would be impossible to implement but which optimizes coding efficiency. Suppose that the probability of a non-redundant sample in any given position of the scan line is $1/N_s$ and is not dependent upon past history. If we were to apply this idea to the previous coding example the probability of a "1" in any given position of the time sequence field would be

$1/N_s$ and the probability of a "0" would be $(1-1/N_s)$. The entropy per bit position within the time sequence field is

$$H = -(1/N_s)\log_2(1/N_s) - (1 - \frac{1}{N_s})\log_2(1 - \frac{1}{N_s}) \qquad (5)$$

Although we do not know a practical way to realize this ideal performance, the existence of a coding scheme is guaranteed which will result in a bandwidth compression given by

$$N_{BW} = \frac{5N_s}{5 + H \cdot N_s} \qquad (6)$$

A plot of N_{BW} versus N_s for optimum coding is shown in Fig.2. A surprising feature of this graph is the fact that the function is very nearly linear over the range $1 \le N_s \le 20$. Over this range the function can be accurately approximated by

$$N_{BW} \simeq 0.562 + 0.438\ N_s \qquad (7)$$

Fractional Bit per Sample Time Coding

One approach which could be employed to improve the efficiency of time sequence coding would be to start with the 500 bit time sequence field discussed on page 12 entitled "Single Bit per Sample Time Tag Coding", and re-encode this data m-bits at a time ($m \ge 2$). Since the 500 bit sequences usually contain far more "0" bits than "1" bits we would encode a single "0" if each bit of the m-bit field is "0". If the entire m-bit field is not zero then we would generate a "1" as a prefix and follow this prefix with a code word which uniquely defines the m-bit pattern. The simplest (but not most efficient) method of performing this coding would be to simply copy the m-bits following the "1" prefix bit. For example, if we employed this scheme with $m = 2$ we would transform the original 500 bit time sequence codes into 250 di-bit fields. This is illustrated below:

Original 500 Bit Sequence:	00	00	01	00	00	00	00	11	10	00	00	00	10
Fractional Bit (M = 2) Coding:	0	0	101	0	0	0	0	111	110	0	0	0	110

If we again assume that the probability of a "1" in the original 500 bit sequence is $1/N_s$ and is independent of past history, the expected number of bits required to provide this time sequence coding can be

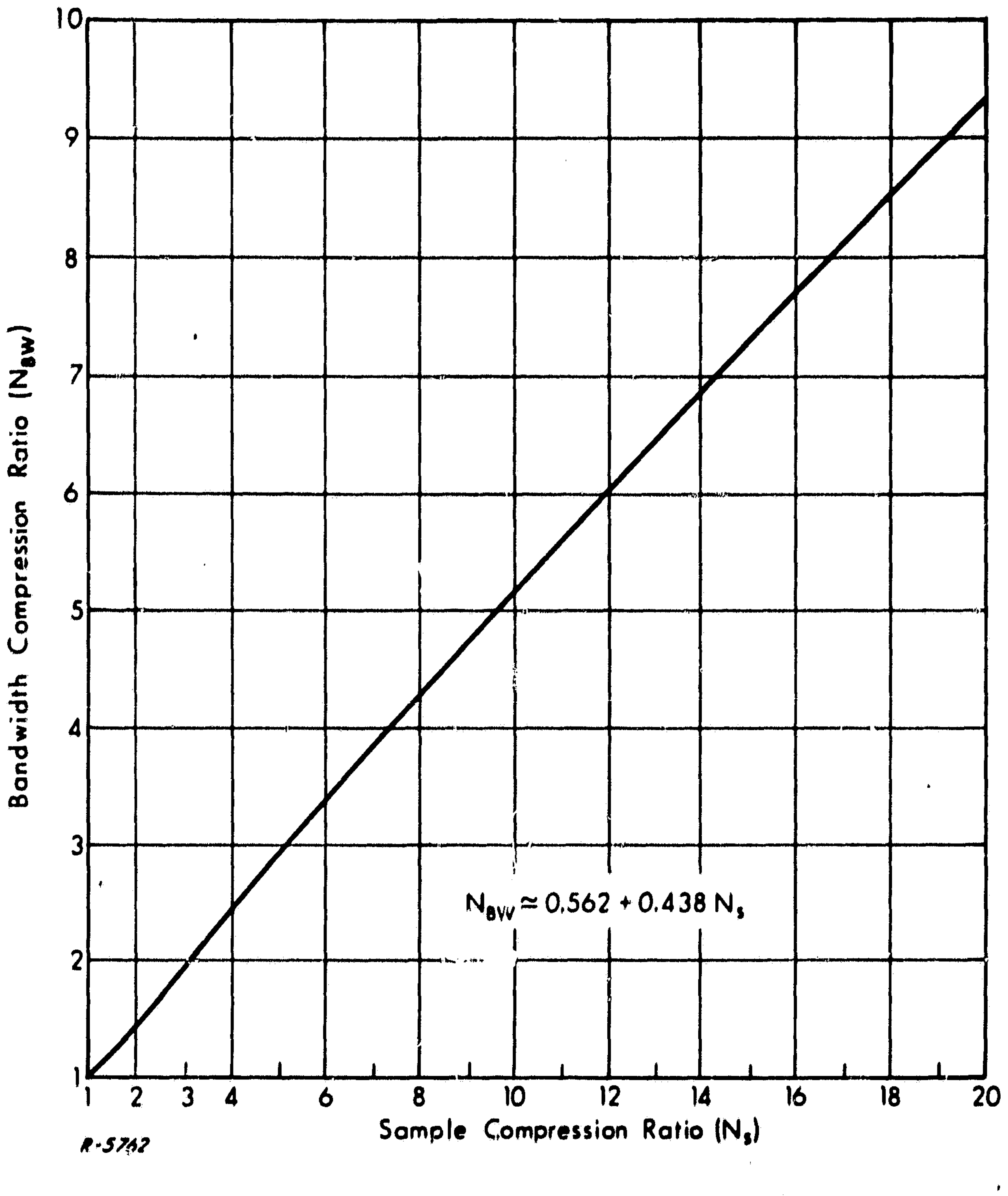


Fig. 2 Plot of N_{BW} versus N_s for Optimum Coding Scheme

shown to be

$$C_T = 500 \{ \frac{1}{m} + [1 - (1 - \frac{1}{N_s})^m] \} \tag{8}$$

Substituting 8) into 3) given

$$N_{BW} = \frac{5N_s}{5 + N_s \{ \frac{1}{m} + [1 - (1 - \frac{1}{N_s})^m] \}} \tag{9}$$

Figure 3 shows the coding efficiency of this fractional bit coding scheme relative to the optimum coding limit. The coding efficiency (γ) has been defined as

$$\gamma = \frac{N_{BW}(\text{fractional bit coding})}{N_{BW}(\text{optimum coding})}$$

The M = 1 curve represents the single bit per sample coding described on page 12 in the section entitled "Single Bit per Sample Time Tag Coding."

Huffman Coding

Additional refinements on this fractional bit coding scheme are possible by applying Huffman coding to the m-bit sequence. In this manner the frequently occurring patterns would be encoded into less than m-bits while the less frequently occurring patterns would require more than m-bits.

Figure 3 illustrates the improvement possible using a variable length code word. The selection of the code words was based on the previously mentioned assumption that the probability of a "0" in any bit position is $1/N_S$, and was optimized for the case where the sample compression ratio (N_s) was 10. The three cases of Huffman coding shown in Fig.4 correspond to the selection of M = 3, 4 and 5. The variation in the length of the code words (including prefix bit) in the three cases is shown below.

No. of Bits in Code Word	M = 3	M = 4	M = 5
Minimum Length	3	3	3
Maximum Length	5	9	12
Average No. of bits to represent a single "1" entry in M-bit field	3	3.25	3.6

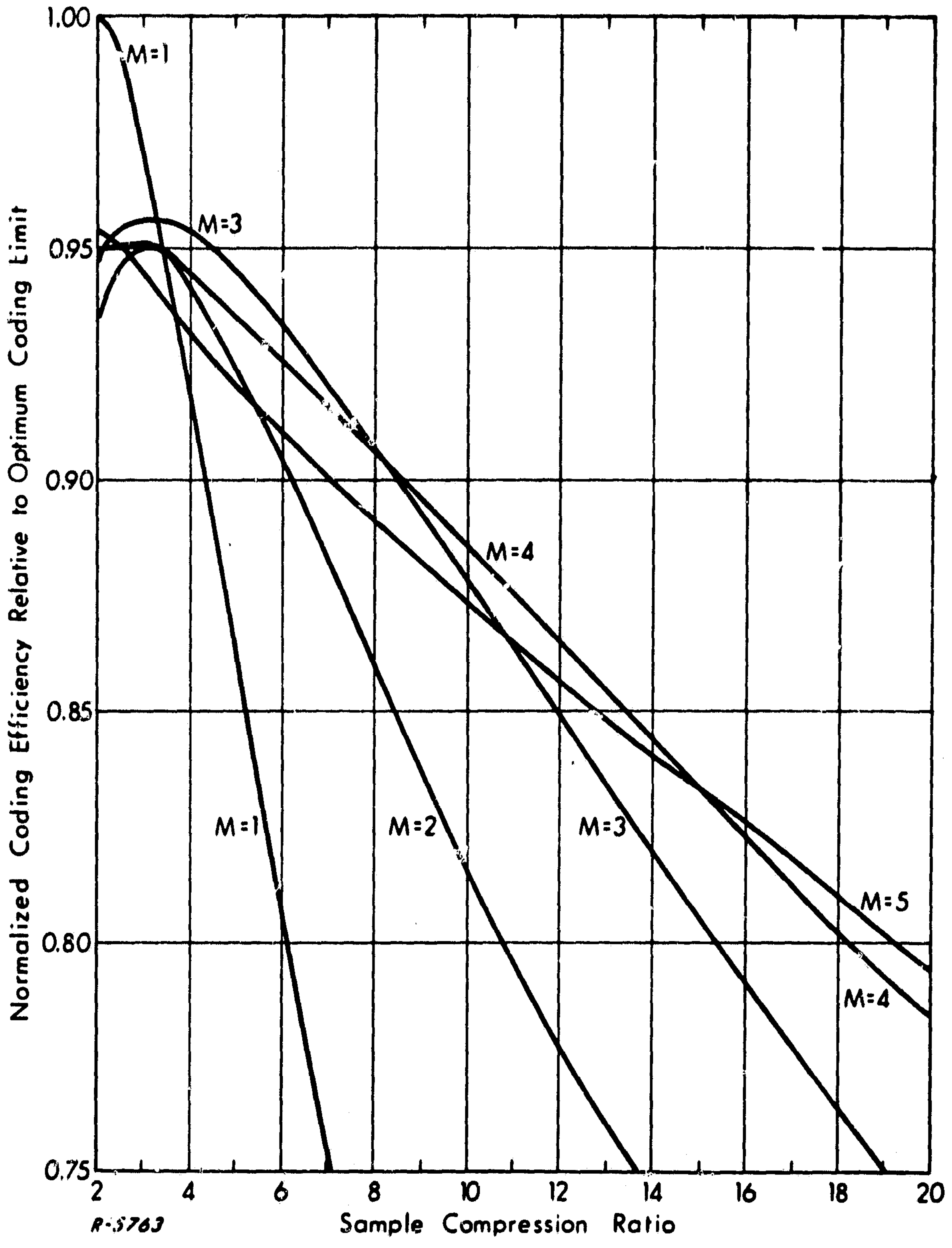


Fig. 3 Coding Efficiency of Fractional Bit Coding Scheme

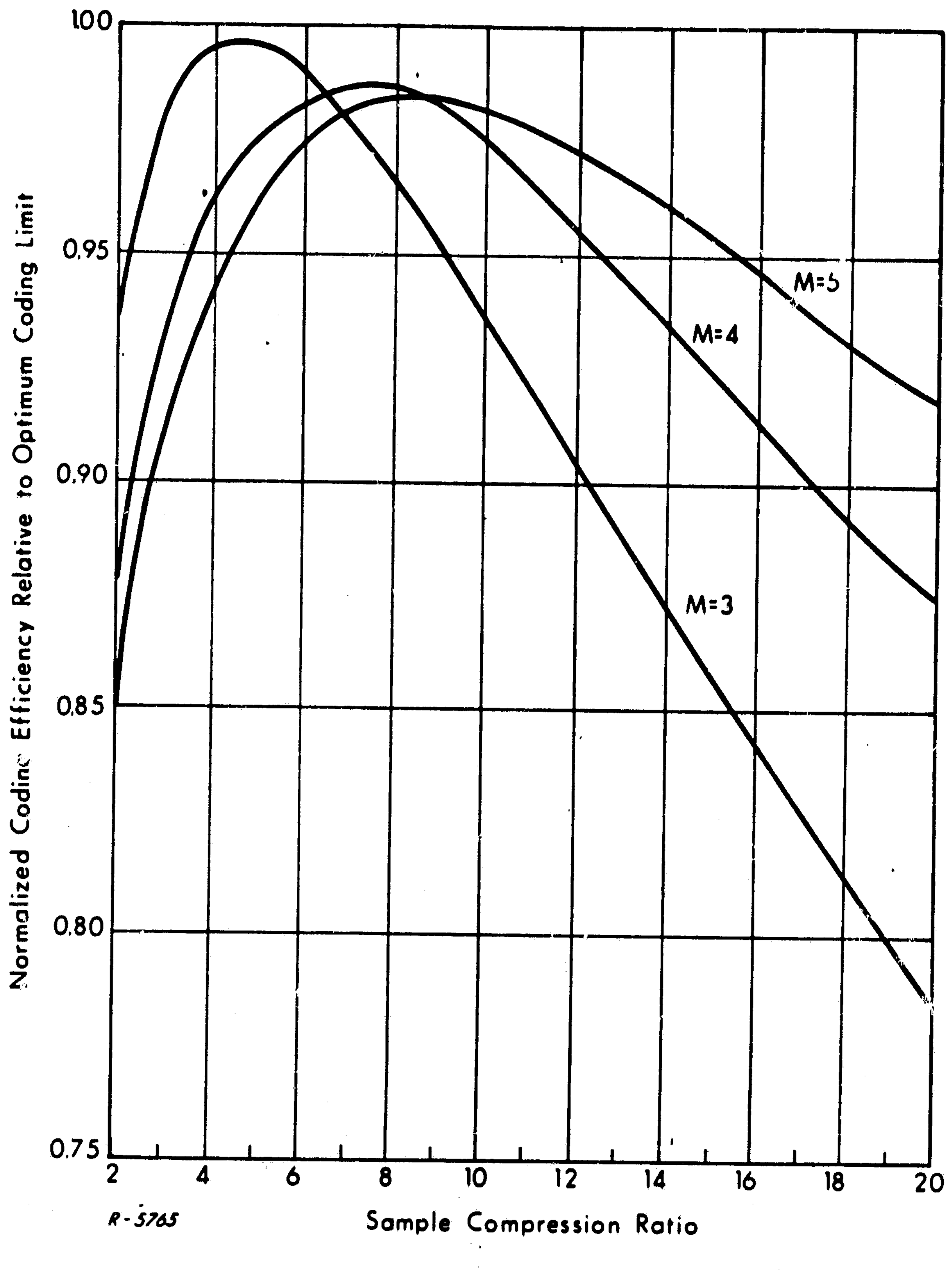


Fig. 4 Huffman-Type Coding Scheme

Normal Run-Length Coding

Another coding scheme which has been extensively used is that of run-length coding. With this scheme only the interval between adjacent non-redundant samples is transmitted. Since the number of samples per scan line is taken to be 500, 9-bits would be required to unambiguously define adjacent run-lengths. If a 9-bit run length field was used

$$N_{BW} = \frac{5N_s}{14} \tag{10}$$

A modified run-length coding scheme is possible which would require less than 9-bits in the run length field. If an m-bit ($m < 9$) run length field is used there is the possibility that a naturally occurring run greater than 2^m will appear. When this happens the coding scheme will have to artificially truncate the run at length 2^m and send an additional compressed data word. The expected number of time tagging bits required under these circumstances are

$$C_T = \frac{500m}{N_s} \sum_{i=1}^{500} \text{INT} \left(1 + \frac{i-1}{2^m} \right) \cdot p(i) \tag{11}$$

where

$$p(i) = \frac{1}{N_s} \left(1 - \frac{1}{N_s} \right)^{i-1}$$

and

$$\text{INT}(k) \equiv \text{Nearest integer} \leq k.$$

The coding efficiency obtained from this normal run length coding scheme is shown in Fig. 5. Over a wide range of sample compression ratios a coding efficiency in excess of 92% can be obtained if a proper choice of the code length is made.

Non-Linear Run-Length Coding

Another type of run length coding scheme has recently been suggested by Bradley in Ref.1 . In this scheme the run length code

[1] Bradley, S. D., "Optimizing a Scheme for Run Length Encoding," Proc. IEEE (Letters), Vol. 57, No. 1, page 108, January 1969.

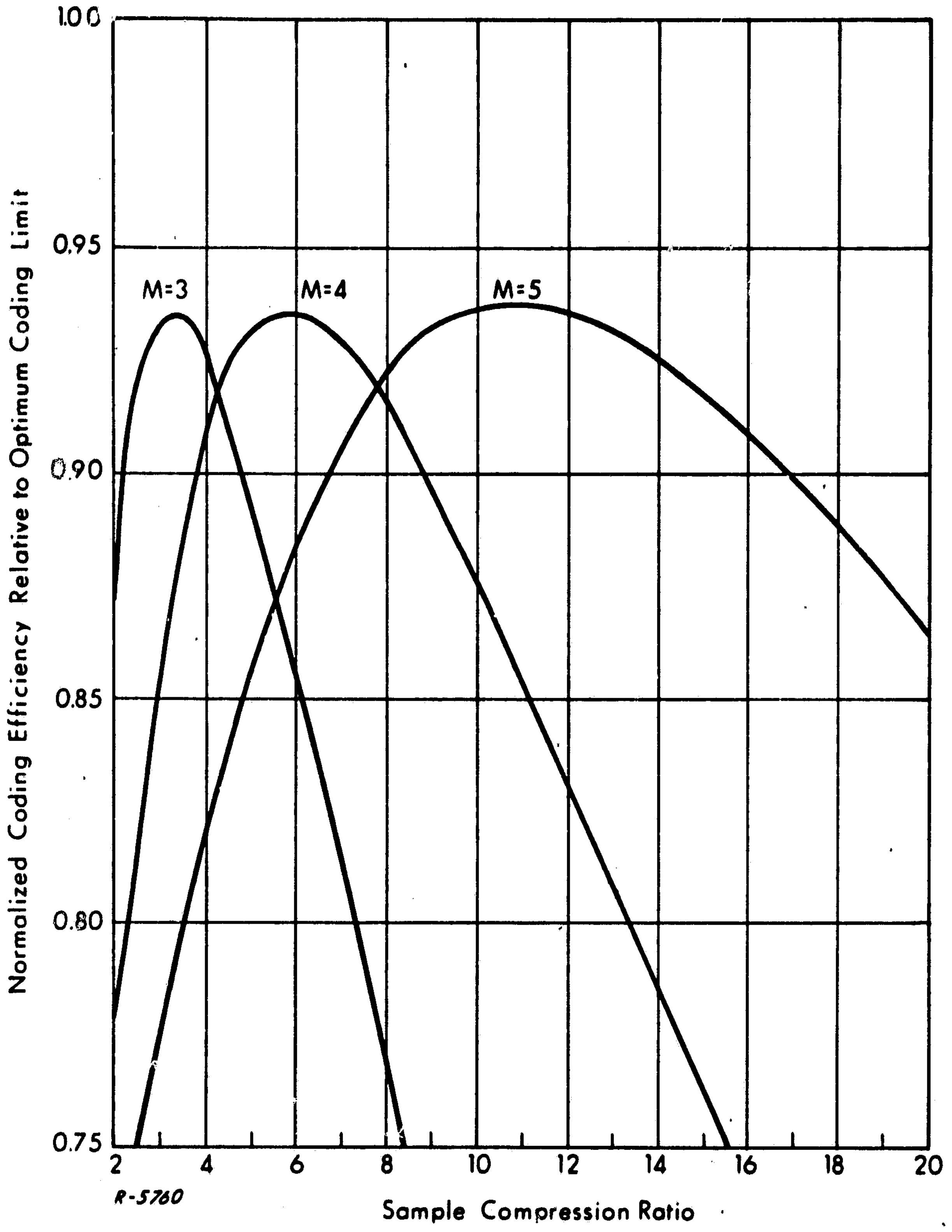


Fig. 5 Coding Efficiency of Normal Run-Length Coding Scheme

words are of two types. The first type indicates a run of a specified duration terminating with a non-redundant sample. In our examples this would be equivalent to a fixed number of zeros (i. e,, redundant samples) followed by a "1" (non-redundant sample). If, because of an excessively long run length, no Type I code exists which will uniquely specify the run, then a Type II code is inserted to be followed by either another Type II code or a Type I code. A Type II code merely indicates the passage of a specified number of "0" bits and it is understood that this does not terminate the run.

As an example, let us consider a 5 bit run length coding scheme having the following characteristics where the codeword index k = 0,1,..31.

Codeword Index	Sample Pattern	
$0 \leq k \leq 27$	$\overbrace{000\ .\ .\ .000}^{k\ \text{zeros}}\ 1$	Type I
k = 28	28 zeros	Type II
k = 29	56 zeros	
k = 30	84 zeros	
k = 31	112 zeros	

All runs in the range $1 \leq$ R.L. ≤ 28 can be expressed with a single Type I code word. All runs between 29 and 140 can be expressed with a single type II code word followed by a single Type I code word. For runs greater than 140, two or more Type II code words would be required.

Three different coding schemes were devised and optimized for a sample compression ratio of 10, using a code length of 3,4 and 5 bits respectively. The maximum run-lengths which can be expressed with a Type I code and also with a single type II code followed by a Type I code is shown below for each case.

Maximum Run Length	M = 3	M = 4	M = 5
1. Type I code alone	5	13	28
2. Type II code followed by Type I code	20	52	140

Figure 6 shows the coding efficiency of the coding schemes shown above. In each case the performance is significantly improved over the normal run-length coding schemes.

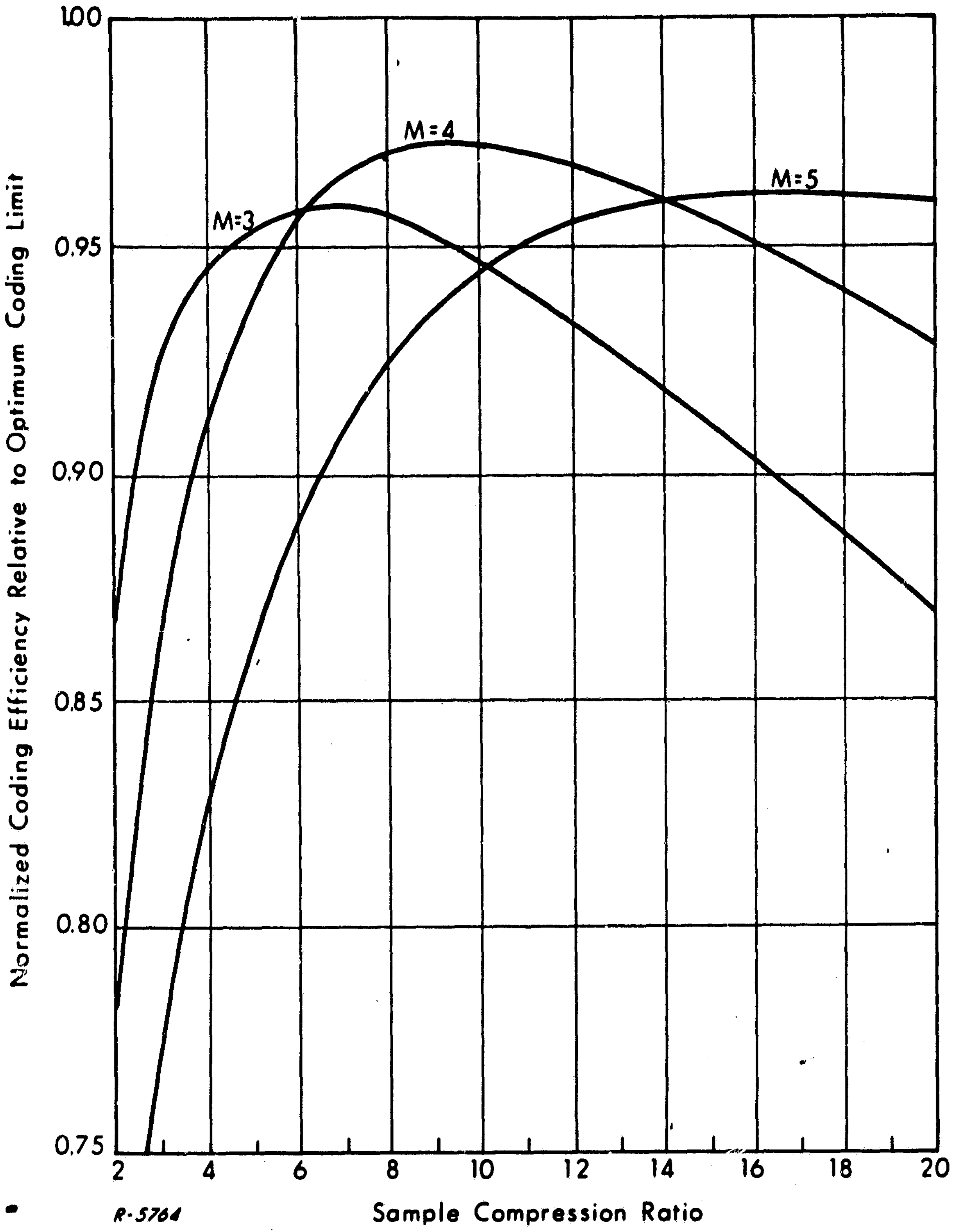


Fig. 6 Coding Efficiency of Non-Linear Run-Length Coding Scheme

Comparisons

A tabulation of the bandwidth compression ratios using the different time sequence coding schemes discussed in this section is shown in Table 1. This table may serve to put into proper perspective the absolute magnitude of the differences between various time sequence coding algorithms.

Figure 7 is a graph of the envelope of the coding efficiency obtained from the four types of coding algorithms discussed in this chapter. For each value of N_s the best choice of "M" is selected for the purpose of plotting the coding efficiency of each algorithm. The vertical lines on the graph show the region in which each parameter selection is optimum for the indicated coding algorithm. It would not be possible to realize the range of performance depicted on this graph with any single choice of the parameter "M".

The two coding algorithms showing the best performance are Huffman coding and nonlinear run-length coding. The envelope of these two algorithms show that Huffman coding is generally superior over the range $1 \le N_s \le 14$ and that nonlinear run length coding is superior for $N_s > 14$.

Past experience with video data compression of planetary landscapes indicates that a sample compression ratio of 10 is quite typical. The three specific selections of coding algorithms and parameters which give the best performance around this region are:

Huffman coding	$M = 5$
Nonlinear run length coding	$M = 4$
Nonlinear run length coding	$M = 5$

Dotted lines on Fig.7 extend the range of performance of these three algorithms. It appears that the Huffman code ($M = 5$) would be the best single choice on the basis of this analysis. Table 2 shows the Huffman code selection for the $M = 5$ case.

Another factor influencing the selection of coding algorithm is the sensitivity of performance in the presence of noise. A more complete discussion of error control will be given in the following chapter. An error occurring within a data field will generally cause a streak of an improper gray shade to occur within the bounds of a single run. This type of error usually will not be particularly objectionable unless the frequency of errors of this type is quite high.

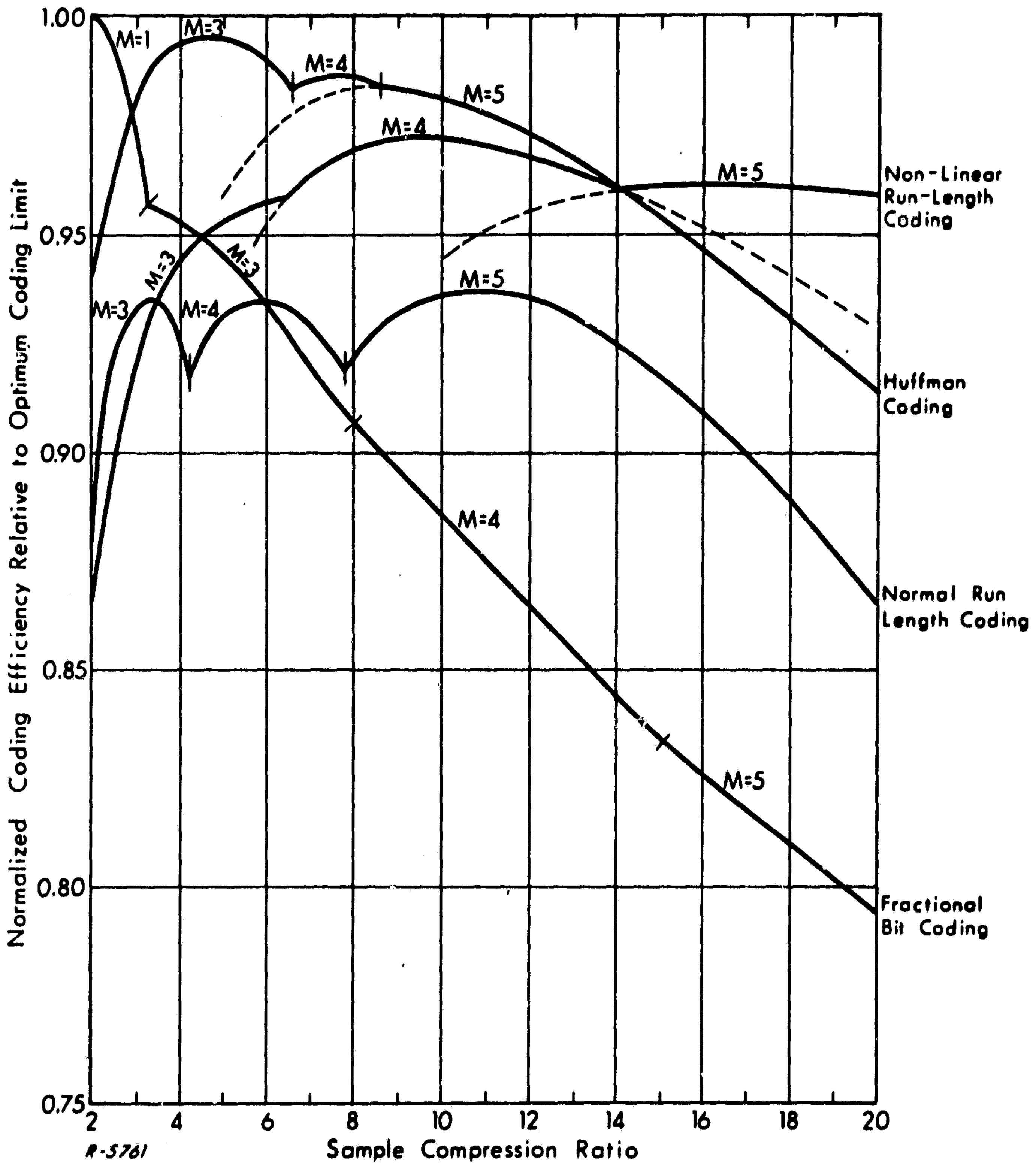


Fig. 7 Comparison of Coding Efficiency Envelope Obtained from Various Coding Schemes

Table 1 Tabulation of Bandwidth Compression Ratios Using Various Time Sequence Coding Schemes

Sample Compression Ratio	Time Sequence Coding Scheme															
	Optimum Coding	Fractional Bit Coding					Huffman Coding			Normal Run Length Coding			Non-Linear Run Length Coding			
		M=1	M=2	M=3	M=4	M=5	M=3	M=4	M=5	M=3	M=4	M=5	M=3	M=4	M=5
1	1.000	0.833	0.769	0.789	0.800	0.806	0.750	0.690	0.694	0.625	0.556	0.500	0.625	0.556	0.500
2	1.429	1.429	1.333	1.348	1.356	1.363	1.333	1.240	1.216	1.245	1.111	1.000	1.236	1.111	1.000
3	1.934	1.875	1.837	1.849	1.839	1.828	1.893	1.794	1.751	1.802	1.664	1.500	1.787	1.664	1.500
4	2.426	2.222	2.286	2.313	2.290	2.260	2.412	2.329	2.276	2.250	2.200	2.000	2.293	2.205	2.000
6	3.371	2.727	3.051	3.149	3.123	3.065	3.340	3.326	3.278	2.878	3.153	2.991	3.230	3.221	2.991
8	4.279	3.077	3.678	3.881	3.880	3.811	4.142	4.223	4.209	3.282	3.920	3.944	4.091	4.153	3.953
10	5.160	3.333	4.202	4.528	4.571	4.506	4.842	5.032	5.069	3.560	4.526	4.828	4.887	5.017	4.872
12	6.021	3.529	4.645	5.103	5.205	5.158	5.457	5.763	5.864	3.761	5.010	5.629	5.626	5.827	5.749
14	6.865	3.684	5.026	5.619	5.789	5.768	6.002	6.427	6.600	3.897	5.401	6.347	6.314	6.591	6.586
16	7.695	3.810	5.356	6.084	6.329	6.343	6.488	7.033	7.282	4.033	5.724	6.986	6.955	7.313	7.393
18	8.513	3.913	5.645	6.505	6.829	6.884	6.924	7.587	7.916	4.129	5.993	7.555	7.553	7.999	8.174
20	9.321	4.000	5.900	6.887	7.294	7.394	7.318	8.095	8.507	4.207	6.221	8.063	8.113	8.651	8.934

Table 2

Huffman Code Selection for M = 5

5 Bit Sequence	Huffman Code	No. of Non-Redundant values in 5 sample sequence
00000	0	0
00001	100	
00010	101	
00100	1100	1
01000	1101	
10000	1110	
00011	11110000	
00101	11110001	
01001	11110010	
10001	11110011	
00110	11110100	2
01010	11110101	
10010	11110110	
01100	11110111	
10100	11111000	
11000	11111001	
00111	11111010	
01011	111110110	
10011	111110111	
01101	111111000	
10101	111111001	3
11001	111111010	
01110	111111011	
10110	111111100	
11010	111111101	
11100	111111110	
01111	11111111100	
10111	11111111101	
11011	111111111100	4
11101	111111111101	
11110	111111111110	
11111	111111111111	5

Errors occurring within a time sequence coding field present a more serious problem since the occurrence of a single error will normally introduce a lateral displacement of the remainder of the scan line by an unpredictable amount. The only coding algorithm discussed in this chapter which is relatively immune to objectional errors of this type is the M = 9 case of normal run length coding. Since 9 bits are reserved for describing the run length it would be possible to specify the end point relative to the start of the scan line rather than using the end point of previous run as the reference. By so doing a single error in a time sequence field would affect only a single run. However, this particular form of time sequence coding is one of the worst schemes which we have considered from the standpoint of coding efficiency.

Among those coding schemes having a high coding efficiency, it appears impossible to state which is to be preferred from the standpoint of noise immunity. Except where complementary errors have been made, it is possible to detect the occurrence of time sequence errors by a simple consistency test. Error correction is, however, a much more difficult task. As will be seen in the next chapter, the additional redundancy which might be added in an attempt to provide this feature may very well cancel the advantage gained by the efficient time sequence coding.

Error Control

One of the most difficult problems in the design of a compressed data system involves the error control subsystem. The effect of uncorrected errors introduced during the transmission of compressed data is much more serious than in the case of uncompressed data. Generally the effect is to distort an entire segment of the data rather than merly introducing an error to a single point. In the case of video data compression a single uncorrected error in a time sequence word often causes a shift in the positioning of the remainder of the scan line.

Fortunately it is fairly easy to detect most errors in a time sequence field of a scan line. When run length coding (either normal or non-linear) is used, this method consists of forcing the output of the last sampled data point in each scan line. At the decoding terminal,errors in the run length fields can usually be detected by observing that the sum of the received run lengths from a given scan line differed from the known number of sampled points along a scan line. All instances in which a single error appears within a run length field of a given scan line will be detected. Most multiple run length errors will also be detected but, in these cases, there is the possibility that complementary errors may occur to prevent detection by this method.

If fractional bit per sample or Huffman time sequence coding is used, an error in a time sequence code field will have one of two effects:

1. If the error results in a code word representing the same number of non-redundant samples in the group, as the proper code word, the error will go undetected but the effect will be merely to distort a very small section (typically 1% of an entire scan line) and therefore to be relatively unobjectionable, and

2. If the error results in a code word representing a different number of non-redundant samples in the group as the proper code word, the error will generally be detectable because the number of data fields within the scan line will not be correct. As with the run length coding, complementary errors can occur to prevent detection. In addition, it is possible that a single bit error could cause the remainder of the scan line to be interpreted incorrectly. However, this will usually result in a detectable error.

One error control technique which has been discussed in our previous reports is that of line substitution. When an error is detected in a time sequence field of a scan line, the decoder replaces the current scan line by the last previously received scan line that was properly decoded. If the scan line rejection ratio is fairly low (say less than 10%), this method results in reconstructed video pictures that are of a high quality on a subjective basis. If the scan line rejection ratio is high this procedure results in noticeable distortions due to lack of resolution and granularity in the vertical dimension.

Another technique which has been previously studied in connection with run length coding scheme is the use of a (7, 4) error correction code on the time sequence field. In 1. a format was analyzed in which the four most significant bits of each 5-bit run length field were encoded with a (7, 4) single error correction code. The least significant bit of the run length field remains unprotected.

1. J. Levy and E.H. Gavenman, "Development of Advanced Techniques for Data Acquisition Processing and Communications," ADCOM Interim Report (NAS12-554), 1 May 1968.

The (7, 4) code insures that if a single error occurs within the encoded 7-bit word, the error will be corrected. Two or more errors within the seven bit field will result in an uncorrectable run length error.

To evaluate the effectiveness of the (7, 4) error correction code a Monto Carlo simulation of channel errors was performed using a modified Pareto channel model. This model was chosen because it simulates, in a fairly realistic manner, the tendency of errors to occur in clusters. This characteristic is typical of most high-quality real communication channels with the exception of the gaussian "space" channel. A description of the modified Pareto model used in this simulation is contained in Appendix B. In this simulation the number of non-redundant data points/scan line was held constant at a value of 45.

Table 3 is a tabulation of the statistics of scan lines containing errors in the encoded run length field that were and were not correctable using the (7, 4) code. The last column lists the percentage of scan lines containing errors in a run length field (s) that are correctable. It appears that this figure is quite insensitive to the choise of clustering factor (α) or long term bit error ratio (ϵ). Roughly three-fourths of all errors in the run length field are correctable with the (7,4) code and the remaining one-fourth are uncorrectable. This excludes those runs made at a long-term error ratio of 10^{-3}, since the Pareto model is generally not valid for channels with this high an error rate.

Despite the fact that the (7, 4) coding will substantially reduce the probability that an uncorrectable error will occur in a run length field, it may be questioned whether the benefits, thus gained, adequately compensate for the resulting decrease in information transmission rate by 23%. Other forms of error correction may prove more effective particularly for burst-error channels. For instance, the time sequence fields from a single scan line could be encoded into a single block code rather than numerous short codes. This will undoubtedly improve the coding efficiency at the price of increasing the complexity of decoding process. However, the encoding complexity would not be significantly increased by this procedure. Another possibility would be for the decoding terminal to detect errors (either by means of the previously discussed consistency check and/or a very simple parity check error detection code) and to perform error correction by correlating the current scan line with the last previously received scan line. The error correction procedure would consist of attempting to determine the positions at which an abrupt change occurred in the corresponding positions of the adjacent

Table 3

Tabulation of Monte Carlo Error Simulation

α	ϵ	Total No. of Scan Lines Simulated	Fraction of Scan Lines with Error (s) in Run Length Fields that are Correctable
.15	10^{-3}	56	0.108
	10^{-4}	175	0.652
	10^{-5}	237	0.759
	10^{-6}	275	0.804
	10^{-7}	286	0.815
	10^{-8}	302	0.867
.2	10^{-3}	106	0.367
	10^{-4}	225	0.716
	10^{-5}	242	0.736
	10^{-6}	258	0.756
	10^{-7}	238	0.765
	10^{-8}	251	0.776
.25	10^{-3}	158	0.569
	10^{-4}	209	0.680
	10^{-5}	197	0.741
	10^{-6}	195	0.708
	10^{-7}	217	0.746
	10^{-8}	213	0.695
Grand Total*		3520	0.748 ± 0.053 (Average and standard deviation)
* Excluding all $\epsilon = 10^{-3}$ runs		29	

scan lines and would attempt to adjust a run length of the current scan line to eliminate the abrupt loss of correlation between the two lines. This procedure has the advantage that no additional redundancy must be added into the message and the message encoding operation is not complicated.

The process of intelligently reducing the inherent redundancy of video data is certainly not simple and a great deal of care is required to achieve a high bandwidth compression ratio without incurring serious distortions in the reconstructed picture. For this reason we should jealously guard any reduction in bandwidth which we have achieved and give up portions of this savings for error control purposes only when such techniques prove their worth.

An appropriate criterion for judging the worth of an error control scheme can be stated in the following terms: For an acceptable level of image quality (fidelity criterion) what is the required channel quality with and without error control. The increased tolerance to channel errors must then be weighed against the increased redundancy. Further work in this area will focus specifically on these questions.

TWO-DIMENSIONAL COMPRESSION ALGORITHMS

One-dimensional (along the scan line) image compression algorithms cannot take advantage of line-to-line correlation except in a limited sense of transmission error control by substituting a valid scan line for an adjacent one in which an error has been detected. It is expected therefore that substantial improvements in compression ratios can be achieved by two-dimensional algorithms. Two approaches for meeting this objective are described below.

Zero-Order Interpolation of 4 x 4 Point Squares

This section derives the two-dimensional compression algorithm and develops a flow chart of the algorithm which can be implemented on a general purpose computer for evaluation. The technique employed involves a two step process. First, the picture is subdivided into 4 x 4 point squares. For each square, one of a small number of pre-defined patterns is selected in accordance with a best-fit criterion. Second, a zero order data compression algorithm is employed for linking the squares together. Before proceeding with the discussions it will be useful to establish some ground rules. First, to minimize processing time and complexity, only four patterns will be used. These patterns are shown in Fig. 8. Each pattern is divided into two equal

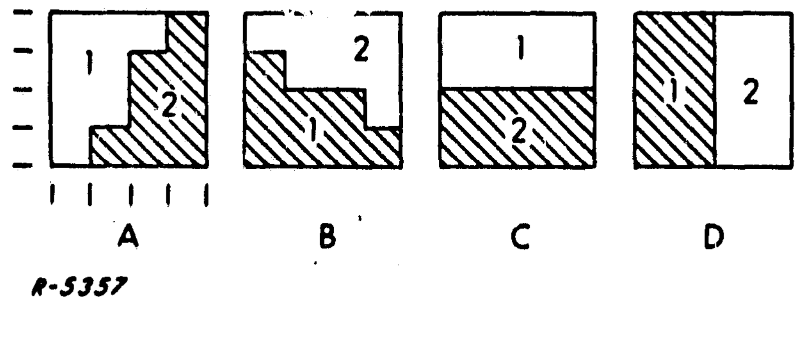


Fig. 8 Standardized Patterns

areas, labeled 1 and 2. Each area can have a different grey tone value. Second, it is assumed that it is not necessary to match the contours exactly. This assumption, when combined with the assumption that only four patterns will be used, should introduce at most a 2 point error in contour position. Finally, it is assumed that the data word need not be a fixed length. This requirement will reduce the total number of bits transmitted since some of the time the grey tone value of both halves of the block will require specification and other times, only one half will require specification. The word format is shown in Fig. 9. Note it is assumed that the picture contains 256 x 256 points. Thus, the horizontal position of each block will be specified as a 6 bit word.

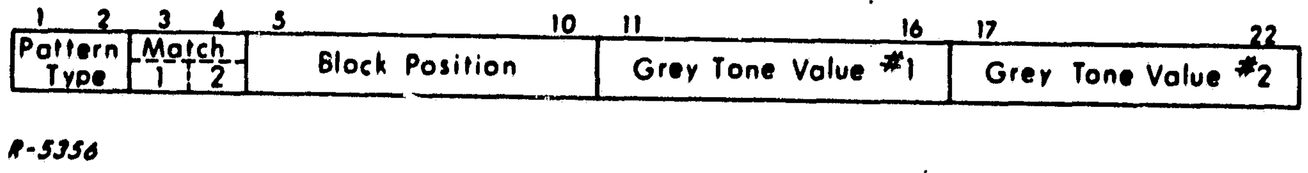


Fig. 9 Data Word Format

Pattern Matching. - The initial step in the procedure is to establish which of the four patterns will be used to represent a particular block. The technique used will select the pattern which represents the actual data with least mean square error. This is accomplished in the following manner.

First, the best fit between each pattern and the existing data must be determined. This is accomplished by determining the average grey tone value within area 1 and area 2 for each pattern. The computed average grey tone values will then be assigned to their appropriate areas. In this manner, each pattern will be fit to the data with minimum mean square error.

Second, the mean square error between the data and the optimized patterns will be computed.

Finally, the pattern with the least mean square error will be selected to represent the block of data.

In this manner, a pattern is selected which most nearly represents the data. The criterion used is best mean square fit. The quality of the pattern match might be improved by a more sophisticated technique, but such techniques would require more complexity.

Zero Order Pattern Extension. - Having selected a pattern to represent the data, it is desirable to see if the selected pattern matches any of its neighbors. If a match is found, the total quantity of data that must be transmitted can be reduced.

Figure 10 shows an internal section of the block pattern. Assume that a pattern has been matched to the square at the i'th row and j'th column. The processing moves from left to right and from top to bottom. It is now necessary to determine if a match exists between block$_{i, j}$ and its neighbors. Which neighbor is matched and the matching requirements, will depend upon the type of block selected. For example, if block D is selected, it will be matched first with block$_{i-1, j}$. If a match, within tolerance for a zero order compressor, is established, then the block need not be transmitted, since no new information is contained in the block. If a horizontal match exists, but no vertical match exists, then new grey tone values are transmitted for both areas.

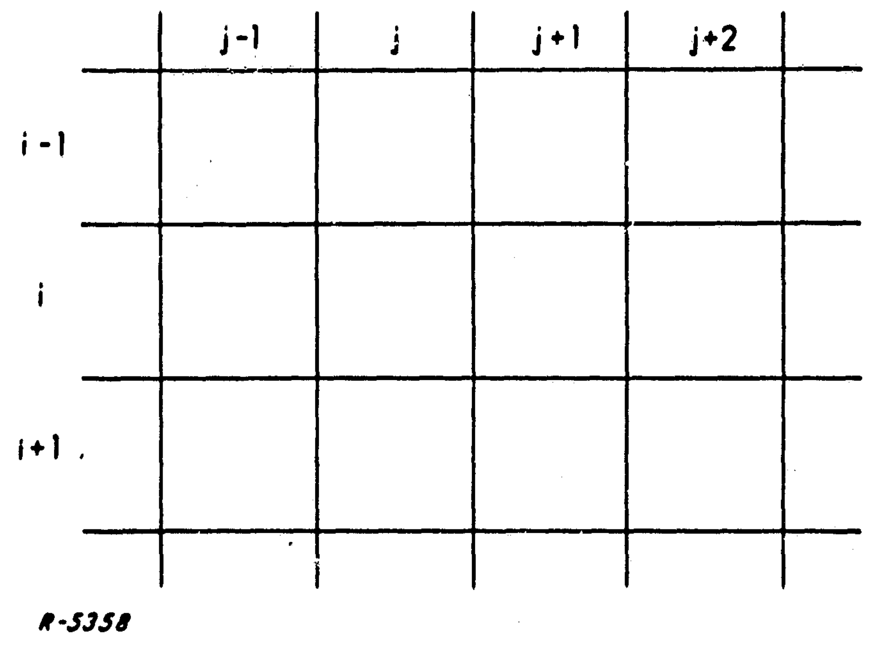


Fig. 10 Internal Section of Block Pattern

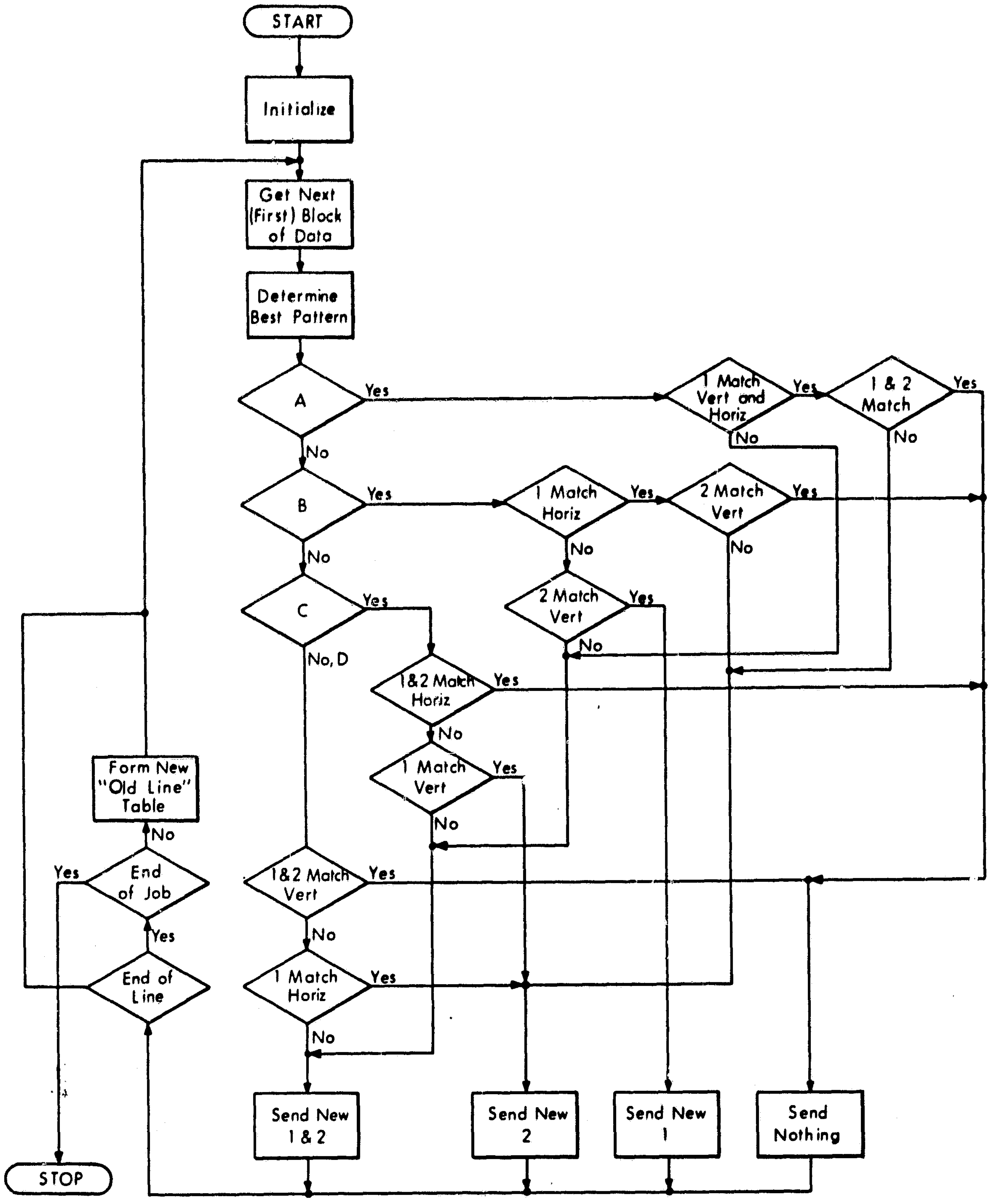
The matching required and decision algorithms for the four standard patterns are presented in Table 4 below.

Table 4

MATCHING AND DECISION ALGORITHMS

Pattern	Match	Decision Algorithm
A	i, j-1 i-1, j against Area 1	No match → send new 1 & 2 match → test match with 2 2 also matches → send no data 2 no match → send new 2
B	i, j-1 against Area 1 i-1, j against Area 2	no match → send new 1 & 2 match both → send no data only 1 match → send new 2 only 2 match → send new 1
C	i, j-1 against Areas 1 & 2 i-1, j against Area 1	match 1 & 2 → no data match 1 and i-1, j → send new 2 no match → send new 1 & 2
D	i-1, j against 1 & 2 i, j-1 against 1	match 1 & 2 → no data match 1 & i, j-1 → send new 2 no match → send new 1 & 2

Computer Algorithm. - From Table 4 it is seen that the decisions to be made are straightforward. With the addition of some processing to handle edge squares, the computer algorithm follows directly from Table 4. The program stores the pattern type and values for the "line" immediately above the one being processed. This means that when no data is transmitted for a particular square, the pattern type and value for that block must still be preserved in the program for future reference. Edge points are handled by assuming that the edges are made up entirely of all white blocks. A flow chart for the algorithm is shown in Fig. 11; program development is underway.



R-5359

Fig. 11 Processing Flow Chart

Compression Within Bounded RMS Error Over a Square

Investigation is continuing into a two-dimensional data compression algorithm which may prove more efficient than those one-dimensional techniques previously studied. This technique is based upon examining a rectangular matrix of raw video data points and attempting to accurately classify the pattern and to extract and transmit the significant parameters in a manner that will permit accurate reconstruction. In the following description a 5 × 5 data point matrix will be used; however, both larger and smaller matrices will be considered. Let the original data point matrix be A where

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix} \tag{1}$$

Initial Conditioning of Matrix. - The compression processor will start by computing the mean value and standard deviation of the elements and recomputing a modified matrix A' of mean zero. Thus

$$\mu = \frac{1}{25} \sum_{i=1}^{5} \sum_{j=1}^{5} a_{ij} \tag{2}$$

$$\sigma^2 = \frac{1}{25} \sum_{i=1}^{5} \sum_{j=1}^{5} (a_{ij} - \mu)^2 \tag{3}$$

$$a'_{ij} = a_{ij} - \mu \tag{4}$$

$$A' = (a'_{ij}) \tag{5}$$

Uniform Matrix Determination. - One type of matrix which will probably occur frequently is that of the uniform matrix (i.e., all matrix elements are identical or nearly identical). This can be easily checked by testing whether

$$\sigma^2 \leq k_u \tag{6}$$

where k_u is some pre-determined constant. If $\sigma^2 \leq k_u$, then the matrix is considered uniform. When this condition is found, the compressor processor attempts to extend the range of the uniform matrix by adding another column of data points. These new data points (in our example a_{16}, a_{26}, a_{36}, a_{46} and a_{56}) are then adjusted using (4) and a new value of σ^2 (but not μ) is recomputed. Again we apply the test given in (6) having omitted column $\{a_{i1}\}$ and if σ is still less than or equal to k_u, another column of data points is processed. This procedure continues until either 1) $\sigma^2 > k_u$ or 2) the number of additional columns processed (after the first five) equal 31. When either condition occurs, an output message is generated containing the following items:

1) Uniform matrix code prefix 2 bits
2) Mean value (μ) 5 bits
3) Number of additional columns processed, N. 5 bits

A uniform matrix code message will consist of 12 bits.

Standard Pattern Correlation. - If the initial test of σ^2 revealed that the modified matrix could not be considered uniform, the compression processor next determines whether the matrix matches one of a set of standard matrices stored within the compressor processor to an acceptable degree of correlation. Let us suppose that there are M pre-stored patterns each one of which is of the form

$$B'_k = (b'_{ij}(k)) \qquad i, j = 1, 2, \ldots, 5 \tag{7}$$

where each of the B' matrices are set to mean zero and the pre-stored patterns are all adjusted such that the variances of each pattern are the same (say $\sigma^2_{B_k} = 1$, $k = 1, 2, 3, \ldots, M$).

Initially a storage register RMAX is reset to zero.

Next the compression processor will compute the correlation function R_k between the modified matrix and each pre-stored pattern, where

$$R_k = \sum_{i=1}^{5} \sum_{j=1}^{5} a'_{ij} \cdot b'_{ij}(k) .$$

Following each trial correlation R_k is compared with RMAX. If $|R_k| >$ RMAX, then $|R_k|$ is gated into the RMAX register, the sign of R_k is gated into a one bit register, RSIGN, and k is gated to KOPT. After all M trial correlations have been completed, RMAX is compared with a pre-stored correlation limit RLIM. If RMAX > RLIM, then the pre-stored pattern which most closely matched the observed matrix is considered to adequately represent the matrix. In this case an output message will be generated containing the following items:

1) Standard pattern prefix 1 bit
2) Number of pattern giving best match (KOPT) 6 bits
3) Mean value of sampled matrix (μ) 5 bits
4) Sign of trial correlation (RSIGN) 1 bit
5) Variance (σ^2). 5 bits

A standard pattern code message will consist of 18 bits.

Note that a high negative correlation of the sample matrix with a pre-stored pattern conveys just as much information as a positive correlation to the same absolute level. Therefore, in reality, two trial correlations are performed with each pattern and item 4) will be necessary to specify the polarity of the best correlation match.

At the data reconstruction end, an estimate of the original sample pattern is generated by 1) accessing the normalized and adjusted pattern which was specified (KOPT) and rescaling the variance to conform to the specified variance, σ^2, and KSIGN, (this, in essence, readjusts the contrasts in the grey scale between the respective matrix elements) and 2) adjusting the rescaled matrix to the specified mean value (μ).

Uncompressed Matrix Transmission. - If the modified matrix can neither be considered uniform nor can be satisfactorily matched with a pre-stored matrix, the compression processor will transmit the matrix elements in uncompressed form. A two-bit prefix code identifying the fact that the matrix will be transmitted in uncompressed format will precede the matrix element readout. An uncompressed matrix code message will consist of 127 bits.

Estimate of Bandwidth Compression Ratio. - It is very easy to compute the bandwidth compression ratio possible for each of the three classifications considered in the preceding sections. In the case of the uniform distribution, a code message may define from four to thirty-five columns of data. Table 5 gives the number of bits/matrix message and the bandwidth compression ratio (for that matrix) for all three classifications.

Table 5
Bandwidth Compression Ratio

Classification	No. of Bits in Original (Uncompressed) Message	No. of Bits in Compressed Message	Bandwidth Compression Ratio
Uniform Matrix (N=0)	125	12	10.41
Uniform Matrix (N=31)	900	12	75.00
Standard Pattern	125	18	6.95
Uncompressible Pattern	125	127	0.985

The overall bandwidth compression ratio will be dependent upon the relative proportion of each classification in the entire video transmission. These figures are not available at this time. In order for this algorithm to be efficient, the fraction of the matrices processed as uncompressible must be kept under 10%. Of course, this classification could be eliminated entirely, in which case the best match with one of the standard patterns (if the uniform matrix test failed) would be used regardless of its absolute value. The advantage of retaining the uncompressed matrix classification is that the maximum RMS error on the reconstructed picture is bounded at some pre-set value. Without this classification there would be no assurance as to the maximum RMS error.

To show the effect of this classification on the bandwidth compression ratio, suppose that 40% of the matrices could be processed as uniform matrices (with N = 4), 50% processed as standard patterns and 10% processed as uncompressed matrices. Under these circumstances the overall effective bandwidth compression ratio is 6.6. Now suppose that the uncompressed matrix classification is eliminated and that the standard pattern proportion is increased to 60%. The bandwidth compression ratio would then increase to 11.2.

The compression processing described above, especially that involved in the standard pattern classification, is quite extensive. One may reasonably be concerned with the problem of whether it is practical to build compression equipment of this variety that can keep up with the transmission rate. To try to acquire some insight into this problem we will start at the output of the satellite telemetry transmitter and work backwards. For our example we will choose 10K bits/sec as the maximum rate one might wish to transmit compressed video data. In many cases it will be necessary to t ansmit at a much lower rate because of signal strength problems. Next we will hypothesize that a bandwidth compression ratio of 10 would be typical using this compression algorithm. This leads us to an input rate to the compressor 100K bits/sec, 20K data points/sec or 800 5x5 data point matrices/sec. Therefore, the time between successive inputs of 5 $\times$ 5 matrices is 1,250 μs.

We turn now to estimating the number of arithmetic, logical operation and memory accesses which must be performed. Let us assume that the standard patterns are stored in a serial memory (such as a long shift register or delay line). Since the patterns are always accessed in the same sequence, this implementation would probably be the least expensive and result in very fast access to the word which is in th accessible position of the memory. ADCOM has built memories and arithmetic processors of this type that can perform a 12 $\times$ 12 bit multiplication and add in 400 μs. Even though this application would only require a 5 $\times$ 5 bit multiplication and add for the trial correlation, we will stick with this figure for our example.

During the initial matrix conditioning the sampled set of data points will be processed and stored in a set of 25 random access registers. This will require approximately 130 μs using typical circuitr . If the matrix cannot be considered uniform, the processor immediately sequences to the standard pattern trial correlations. Each trial correlation requires 25 multiplications and additions as well as a few one-time-only initialization and testing steps (estimated at 2.5 μs/trial correlation). This results in a total processing time/trial correlation of 12.5 μs.

The total time required to process completely one 5 × 5 matrix of data points would be

$$T = 130 + 12.5\ M\ \mu s\ ,$$

where M is the number of trial correlations. Since the available time per 5 × 5 matrix is 1,250 μs, this means that 76 trial correlations would be possible. For ease of implementation it would probably be best to round this off to the next lower power of two.

The above calculations are not intended to be definitive but merely to indicate that even with a conservative set of assumptions it would be practical to develop such a two-dimensional data compressor algorithm which would operate at an input rate of 100,000 bits/sec and would perform up to 64 trial correlations on each matrix.

PRECEDING PAGE BLANK NOT FILMED.

APPENDIX A

PROGRAM LISTINGS

SUBROUTINE XTLR LISTING

```
C
C         SUBROUTINE TO COUNT TOLERANCE ERRORS
C
      SUBROUTINE XTLR (MR,LA,LB,N,M,MAX)
      DIMENSION MR(N),LA(N),LB(N)
      IER=0
      DO 30 I=1+M,N-M
      DO 10 L=-M,M
      FF=LA(I+L)-LB(I)
      IF (ABS(FF).LE.MAX) IF (IER) 20,30,20
   10 CONTINUE
      IER=IER+1
      GO TO 30
   20 MR(IER)=MR(IER)+1
      IER=0
   30 CONTINUE
      RETURN
      END
```

SUBROUTINE AREA LISTING

```
C
C         SUBROUTINE FOR VIDEO AREA STATISTICS
C
      SUBROUTINE AREA (MN,MATRIX,N1,N2,LINE,N,
     +MA,MB,MC,MD,M,LARGE,N3,Y)
      DIMENSION MATRIX(N1,N2),LINE(N),LARGE(N3,2)
      DIMENSION MA(M,3),MB(M,3),MC(M),MD(M)
C
C         ASSEMBLE NEW WORKING MATRIX FROM LINE
C
      I=0
      J=1
   10 IF (LINE(J+1).EQ.LINE(J)) GO TO 30
   20 I=I+1
      IF (I.GT.M) PAUSE 'ERROR: WORKING MATRIX DIMENSIONS.'
      MA(I,1)=J
      MA(I,2)=LINE(J)+1
   30 J=J+1
      IF (J-N) 10,20,40
   40 ISAVE=I
      I=1
      K=1
      ITER=ITER+1
      IF (ITER.NE.1) GO TO 100
```

```
C
C          COMPUTE OUTPUT MATRIX SWITCH
C
      MIN=100
      ASSIGN 530 TO NEXT
      IF (N1.GT.100) GO TO 41
      MIN=N1
      ASSIGN 560 TO NEXT
      GO TO 42
   41 X=N1-100
      Z=(ALOG(100.))/X
      Y=EXP(Z)
C
C          FIRST ITERATION LOCATOR LIST
C
   42 DO 50 L=1,ISAVE
      MA(L,3)=L
      MD(L)=1
   50 CONTINUE
      GO TO 70
C
C          INCREMENT ALL AREAS
C
   60 IF (MB(1,3).EQ.0) GO TO 61
      IA=MB(1,3)
      MC(IA)=MC(IA)+MB(1,1)
   61 DO 62 L=2,KSAVE
      IF (MB(L,3).EQ.0) GO TO 62
      IA=MB(L,3)
      MC(IA)=MC(IA)+MB(L,1)-MB(L-1,1)
   62 CONTINUE
C
C          TRANSFER MA TO MB
C
   70 DO 72 J=1,3
      DO 71 L=1,M
      MB(L,J)=MA(L,J)
      MA(L,J)=0
   71 CONTINUE
   72 CONTINUE
      KSAVE=ISAVE
      IF (ITER.EQ.MN) GO TO 80
      RETURN
C
C          LAST ITERATION CLOSE OUT
C
   80 ITER=0
      K=0
      KOUT=1
   85 K=K+1
      GO TO 500
   90 IF (K.LT.KSAVE) GO TO 85
      LSAVE=0
     RETURN
```

```
C
C         START TEST FOR CONTINUITY AND VALUE
C
  100 ILINK=0
      KLINK=0
  110 IF ((I.GT.ISAVE).OR.(K.GT.KSAVE)) GO TO 60
      IF (MA(I,1)-MB(K,1)) 280,200,120
C
C         MA SEGMENT ENDS AFTER MB SEGMENT
C
  120 IF (MA(I,2).EQ.MB(K,2)) GO TO 150
      IF (KLINK.EQ.K) GO TO 140
      KOUT=2
      GO TO 500
  140 K=K+1
      GO TO 110
  150 KONT=1
      GO TO 400
  160 ILINK=I
      GO TO 140
C
C         MA SEGMENT ENDS WITH MB SEGMENT
C
  200 IF (MA(I,2).EQ.MB(K,2)) GO TO 220
      K=K+1
      LOOP=1
      GO TO 280
  210 K=K-1
      LOOP=0
      GO TO 120
  220 KONT=2
      GO TO 400
  230 I=I+1
      K=K+1
      GO TO 110
C
C         MA SEGMENT ENDS BEFORE MB SEGMENT
C
  280 IF (MA(I,2).EQ.MB(K,2)) GO TO 310
      IF (ILINK.EQ.I) GO TO 300
      DO 290 J=1,M
      IF (MD(J).EQ.0) GO TO 291
  290 CONTINUE
      PAUSE 'ERROR: WORKING MATRIX DIMENSIONS.'
  291 MD(J)=1
      MA(I,3)=J
  300 I=I+1
      IF (LOOP.EQ.1) GO TO 210
      GO TO 110
  310 KONT=3
      GO TO 400
  320 KLINK=K
      GO TO 300
```

```
C
C         SUB-SUBROUTINE TO ESTABLISH CONTINUITY
C
  400 IA=MA(I,3)
      KA=MB(K,3)
      IF (KLINK.EQ.K) MD(KA)=MD(KA)+1
      IF (ILINK.EQ.I) GO TO 410
      MA(I,3)=KA
      GO TO (160,230,320), KONT
  410 MD(IA)=MD(IA)-1
      IF (IA.NE.KA) GO TO 420
      GO TO (160,230,320), KONT
  420 DO 430 J=K,KSAVE
      IF (MB(J,3).NE.KA) GO TO 430
      MB(J,3)=IA
      MD(IA)=MD(IA)+1
      MD(KA)=MD(KA)-1
      IF (MD(KA).EQ.0) GO TO 440
  430 CONTINUE
  440 MC(IA)=MC(IA)+MC(KA)
      MC(KA)=0
      GO TO (160,230,320), KONT
C
C         SUB-SUBROUTINE TO OUTPUT AREA STATISTIC
C
  500 KSTART=0
      KA=MB(K,3)
      MB(K,3)=0
      IF (K.GT.1) KSTART=MB(K-1,1)
      MC(KA)=MC(KA)+MB(K,1)-KSTART
  510 IF (MD(KA).EQ.1) GO TO 520
      MD(KA)=MD(KA)-1
      GO TO (90,140), KOUT
  520 JV=MB(K,2)
      IF (MC(KA).LE.MIN) GO TO 540
      GO TO NEXT
  530 IF (MC(KA).GT.10000) GO TO 560
      FF=MC(KA)
      JA=(ALOG(FF/100.))/Z
      JA=JA+101
      GO TO 550
  540 JA=MC(KA)
  550 MATRIX(JA,JV)=MATRIX(JA,JV)+1
      GO TO 570
  560 LSAVE=LSAVE+1
      IF (LSAVE.GT.N3) PAUSE 'LARGE AREA OVERFLOW'
      LARGE(LSAVE,1)=MB(K,2)-1
      LARGE(LSAVE,2)=MC(KA)
  570 MC(KA)=0
      MD(KA)=0
      GO TO (90,140), KOUT
      END
```

PRECEDING PAGE BLANK NOT FILMED.

APPENDIX B

MODIFIED PARETO CHANNEL MODEL

Berger and Mandelbrot[1] were the first to apply a Pareto distribution as a model for predicting the error characteristics of digital data transmitted over telephone lines. Since then Sussman[2] and others have shown that the general form of the Pareto distribution can often be used to model the error statistics obtained from a wide range of real communication channels. One important characteristic of various communication channels is the tendency for error to occur in bursts. This feature can be realistically simulated by the Pareto model.

The defining statistic of the normal Pareto model is

$$g(t) = \text{Prob}(\text{gap} = t) = \alpha \cdot t^{-\alpha-1} \qquad \text{(A-1)}$$

where α is the Pareto clustering factor $(0 < \alpha < 1)$ and the gap is defined to be the number of bit positions since the last previous bit error.

One problem with the above probability density function is that it results in an infinite mean gap length and consequently a long term error rate of zero. For this reason a modified Pareto distribution is proposed such that the probability density function of the gap length is defined by

$$g(t) = \text{Prob}(\text{gap} = t) = \frac{\alpha \cdot k_o \cdot t^{-\alpha-1}}{1 + k_1 t} \qquad \text{(A-2)}$$

1. J. M. Berger and B. Mandelbrot, "A New Model for Error Clustering in Telephone Circuits," IBM Journal, July 1963, pp. 224-236.

2. S. M. Sussman, "Analysis of the Pareto Model for Error Statistics on Telephone Circuits," IEEE Trans. on Comm. Theory, June 1963, pp. 213-221.

The values of k_o and k_1 are determined by two conditions:

$$\int_1^\infty g(t)dt = 1 \qquad \text{(Normalization of distribution)} \qquad \text{(A-3)}$$

and

$$\int_1^\infty t \cdot g(t)dt = \frac{1}{\epsilon} \qquad \text{(Condition for adjusting)} \qquad \text{(A-4)}$$

distribution to conform to empirically determined long term bit error rate, ϵ.

Evaluating (A-3) by parts gives

$$\int_1^\infty g(t) \simeq \frac{k_o}{1 + k_1} \qquad \text{(A-5)}$$

Thus

$$k_o = 1 + k_1 \qquad \text{(A-6)}$$

Now evaluating (A-4) we have

$$\int_1^\infty t \cdot g(t)dt = \alpha \cdot (1 + k_1)k_1^{\alpha-1}\left[B(1 - \alpha,\alpha) - \frac{k_1^{1-\alpha}}{1 - \alpha}\right] \qquad \text{(A-7)}$$

The term $\frac{k_1^{1-\alpha}}{1-\alpha}$ is negligible compared with $B(1 - \alpha,\alpha)$ and may be dropped. Solving for k_1 between (A-4) and (A-7) gives

$$k_1 = [\epsilon \cdot \alpha \cdot B(1 - \alpha,\alpha)] \qquad \text{(A-8)}$$

where B(m, n) is the Beta function.

In summary, the modified Pareto distribution is defined by

$$g(t) = \text{Prob}(\text{gap} = t) = \frac{\alpha \cdot (1 + k_1) \cdot t^{-\alpha-1}}{1 + k_1 t} \qquad \text{(A-9)}$$

where k_1 is defined by (A-8). The two independent parameters of this distribution are α, the Pareto clustering factor, and ϵ, the long term bit error rate.

APPENDIX C

NEW TECHNOLOGY

After a diligent review of the work performed under this contract, no new innovation, discovery, improvement or invention was made.