GODDARD GRANT

IN-32-CR

111211

27P.

Semi-Annual Status Report

Design of Source Coders and Joint Source/Channel Coders for Noisy Channels

(NASA Grant NAG 5-916)

Submitted to

Instrument Division

Engineering Directorate

Goddard Space Flight Center

Greenbelt, Maryland 20771

ATTN: Dr. Warner Miller (Code 728.4)

Principal Investigator

Dr. Khalid Sayood

Research Assistants

Martin C. Rost

Alan Michels

Department of Electrical Engineering

University of Nebraska-Lincoln

Lincoln, Nebraska 68588-0511

Research Period Reported

May 15, 1987 – November 15, 1987

Introduction

The research conducted during this six month period can be divided into three areas. In all these three areas significant results were obtained. Specifically,

1. (a) The theory behind the proposed joint source/channel coding approach was developed. This has provided insights into the design of robust source coders.

   (b) A variable rate design approach which provides substantial improvement over current joint source/channel coder designs was obtained.

2. (a) The Rice algorithm was evaluated and its advantages and shortcomings were examined in detail.

   (b) An alternative algorithm was obtained which outperforms the Rice algorithm both in terms of data compression and noisy channel performance.

3. A high fidelity low rate image compression algorithm was developed which provides almost distortionless compression of high resolution images.

Section 1

Design of Joint Source/Channel Coders

1.1 Motivation

A block diagram of a typical communication system is shown below.

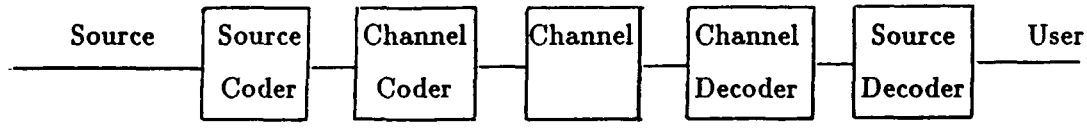

Figure 1.1: A typical communication system

The source coder removes redundancy from the input thus reducing the amount of information to be transmitted. Redundancy is reintroduced in a "controlled" manner by the channel coder. By "controlled" manner we mean the redundancy introduced is of a form which can be used by the channel decoder. The source coder is generally designed without taking the channel statistics into consideration. Similarly the channel coder is designed without consideration of the source statistics. This separation of source and channel coder design is justified by a result of Shannons. Shannon [1] has shown that when the rate of transmission R is less than the channel capacity C, there exists coding schemes which allow us to drive the probability of error arbitrarily close to zero. In this situation the link between the source encoder and decoder is essentially error free. As such, the source coder/decoder pair can be designed without any regard for the effect of noise on the decoding process. Also, if the source coder output contains no redundancy, it can be viewed as samples of an iid process. Thus the channel coder/decoder can be designed without taking into account the source coder output statistics.

These separation arguments break down if, for whatever reason, either of the following happens.

1. The input to the source decoder is different from the output of the source encoder, i.e., the link between the source encoder/decoder is no longer error free.

2. The output of the source coder contains redundancy.

If (1) occurs, the effect of channel error on the decoding process needs to be considered. In this situation the source encoder/decoder pair design should be such as to minimize the effect of channel errors. This situation has been studied by various researchers [2]-[7]. The situation (2) occurs under a variety of situations. Incorrect assumptions about the statistical parameter describing the source, results in correlation in the source output [8]. Non-stationarity of the source may also cause the appearance of redundancy. This redundancy can be used to correct errors in the channel [9]-[12]. The source coders which mitigate the effect of channel errors are collectively known as joint source/channel coders.

In this paper we present an approach to joint source/channel coder design. To facilitate the presentation some nomenclature is in order. First we redraw our block diagram.

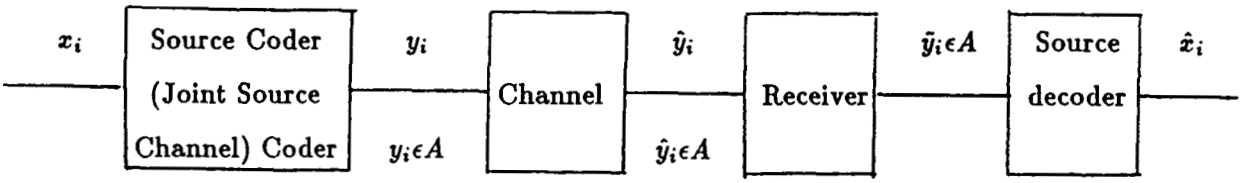

Figure 1.2: Proposed system

Note that the channel coder has been removed while the channel decoder has been replaced by a marked receiver. The purpose of the receiver is to use the redundancy at the ouput of the source coder to provide error protection.

Let the source coder alphabet be denoted by A where

$$A = \{a_1, a_2, \ldots, a_M\}$$

The source coder output is denoted by y_i, the channel output by $\hat{y}_i$, and the receiver output by $\tilde{y}_i$. We go about designing the receiver in the following manner. Recall that in the classical formulation, the optimal receiver (in terms of maximizing the probability of correct decision) is one which selects a_k to maximize

$$P[y_i = a_k | \hat{y}_i]$$

We extend this formulation to decoding sequences of received symbols rather than one symbol at a time. To this end we define

$$\mathbf{y} = (y_0, y_1, y_2, \cdots, y_L)$$

$$\hat{\mathbf{y}} = (\hat{y}_0, \hat{y}_1, \hat{y}_2, \cdots, \hat{y}_L)$$

Furthermore, we assume that the first coder output symbol y_0 is known. This can be justified by assuming that the first output symbol represents the coder output when the source is quiescent. Thus the optimum receiver will maximize

$$P[\mathbf{y}/\hat{\mathbf{y}}] = P[y_0, y, , \ldots, y_L | \hat{y}_0, \ldots, \hat{y}_L] \tag{1.1}$$

This can be rewritten as

$$\begin{aligned} P[\mathbf{y}/\hat{\mathbf{y}}] = P[y_L | y_{L-1}, \ldots, y_0, \hat{y}_0, \ldots, \hat{y}_L] \cdot P[y_{L-1} | y_{L-2}, \ldots, y_0, \hat{y}_L] \cdot \\ P[y_{L-2} | y_{L-3}, \ldots, y_0, \hat{y}_0, \ldots, \hat{y}_L] \cdots P[y_1 | y_0, \hat{y}_0, \ldots, \hat{y}_L] \\ \cdot P[y_0 | \hat{y}_0, \ldots, \hat{y}_L] \end{aligned} \tag{1.2}$$

Assume that given y_{n-1}, y_n is conditionally independent of y_{n-k} $k > 1$. Then, noting that the last factor in (1.2) is unity, and assuming a memoryless channel

$$P[\mathbf{y}|\hat{\mathbf{y}}] = P[y_2 | y_{L-1}, \hat{y}_L)] P[(y_{L-1} | y_{L-2}, \hat{y}_{L-1}] \cdots P[y_1 | y_0, \hat{y}_1] = \Pi_{i=1}^{L} P[y_i | y_{i-1}, \hat{y}_i]$$

Maximizing $P[\mathbf{y}|\hat{\mathbf{y}}]$ is the same as maximizing its log. Thus the optimum receiver maximizes

$$\log P[\mathbf{y}|\hat{\mathbf{y}}] = \sum \log P[y_i | y_{i-1}, \hat{y}_i]$$

If we call $\log P[\mathbf{y}|\hat{\mathbf{y}}]$ the path metric then $\log P[y_i | y_{i-1}, \hat{y}_i]$ is the branch metric. The design of a receiver which maximizes a path metric which is the sum of branch metrics is a well known problem in several different fields. In the field of communications this is simply the problem of design of decoders for convolutional codes.

While the structure of the receiver is evident, we still have to obtain the values of $P[\mathbf{y}_i/\mathbf{y}_{i-1}, \hat{y}_i]$. Sayood and Borkenhagen [9] have obtained an expression for $P[y_i | y_{i-1}, \hat{y}_i]$ in terms of the source coder output transition probabilities $P[y_i | y_{i-1}]$ and the channel transition probabilities $P[\hat{y}_i | y_i]$ as

$$P[y_i = a_j | y_{i-1} = a_m, \hat{y}_i = a_n] = \frac{P[\hat{y}_i = a_n | y_i = a_j] P[y_i = a_j | y_{i-1} = a_m]}{\sum_l P[y_i = a_l | y_{i-1} = a_m] P[\hat{y}_i = a_n | y_i = a_l]}$$

For implementation, the channel transition probabilities can be obtained by modeling the channel as a DMC. The source coder output transition probabilities can be obtained from training sequences.

This particular approach toward using the redundancy in the source coder output is especially attractive as it leaves the door open to other joint source channel coding approaches.

This approach was used by Sayood and Borkenhagen [9] in the differential encoding of images. The results were highly satisfying.

1.2 Design

The main error correcting power of this method arises due to the variations in the source coder output conditional probabilities. To see this more clearly let us examine the conditions under which an error is made. Referring to Figure 1.3, assume that the correct sequence of transmitted codewords was $a_o a_o a_o$. An error occurs if the path $a_o a_j a_o$ is selected over the path $a_o a_o a_o$. This will happen if the path metric for $a_o a_j a_o$ is greater than the path metric of $a_o a_o a_o$. Assume $\hat{y}_1 = a_n, \hat{y}_2 = a_m$.

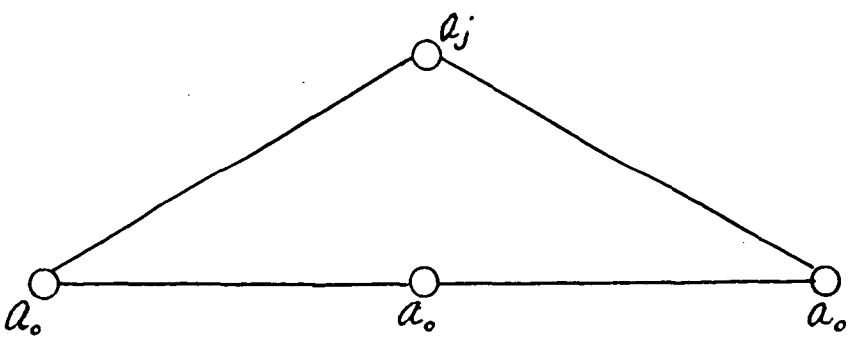

Figure 1.3: Alternate paths at the receiver

Then an error occurs if the quantity

$$\log \frac{P[\hat{y}_1 = a_n | y_1 = a_j] P[y_i = a_j | y_o = a_o]}{\sum_l P[y_1 = a_l | y_o = a_o] P[\hat{y}_i = a_n | y_1 = a_l]}$$
$$+ \log \frac{P[\hat{y}_2 = a_m | y_2 = a_o] P[y_2 = a_o | y_1 = a_j]}{\sum_l P[y_2 = a_l | y_1 = a_j] P[\tilde{y}_2 = a_m | y_2 = a_l]}$$

$$
\begin{aligned}
&- \log \frac{P[\hat{y}_1 = a_n | y_o = a_o] P[y_1 = a_o | y_o = a_o]}{\sum_l P[y_1 = a_l | y_o = a_o] P[\hat{y}_1 = a_n | y_1 = a_l]} \\
&+ \log \frac{P[\hat{y}_2 = a_m | y_2 = a_o] P[y_2 = a_o | y_1 = a_o]}{\sum_l P[y_2 = a_l | y_1 = a_o] P[\hat{y}_2 = a_m | y_2 = a_l]}
\end{aligned} \tag{1.3}
$$

is greater than zero. Cancelling the indicated terms and rearranging terms we get

$$
\begin{aligned}
\log \frac{P[\hat{y}_1 = a_n | y_1 = a_j]}{P[\hat{y}_1 = a_n | y_1 = a_o]} + \log \frac{P[y_1 = a_j | y_0 = a_o]}{P[y_1 = a_o | y_o = a_o]} + \log \frac{P[y_2 = a_o | y_1 = a_j]}{P[y_2 = a_o | y_1 = a_o]} \\
- \log \frac{\sum_l P[y_2 = a_l | y_1 = a_j] P[\hat{y}_2 = a_m | y_2 = a_l}{\sum_l P[y_2 = a_l | y_1 = a_j] P[\hat{y}_2 = a_m | y_2 = a_l]}
\end{aligned}
$$

Assuming the channel to be a binary symmetric channel and $W = \log_2 M$, the length of the codewords

$$
P[\hat{y}_i = a_n | y_1 = a_j] = p^{d_{nj}} (1-p)^{W - d_{nj}} = \left(\frac{p}{1-p} \right)^{d_{nj}} (1-p)^W
$$

where d_{ij} is the Hamming distance between a_i and a_j and p is the channel crossover (bit error) probability. Then

$$
\log \frac{P[\hat{y}_1 = a_n | y_1 = a_j]}{P[\hat{y}_1 = a_n | y_1 = a_o]} = \log \frac{p^{d_{nj}} (1-p)^{W - d_{nj}}}{p^{d_{no}} (1-p)^{W - d_{no}}} = (d_{nj} - d_{no}) \log \frac{p}{1-p} \tag{1.4}
$$

define

$$
g_{lk} = P[y_i = a_l | y_{i-1} = a_k]
$$

Then an error occurs if

$$
(d_{nj} - d_{no}) \log \left(\frac{p}{1-p} \right) + \log \frac{g_{jo}}{g_{oo}} + \log \frac{g_{oj}}{g_{oo}} - \log \frac{\sum_l (\frac{p}{1-p})^{d_{ml}} g_{lj}}{\sum_l (\frac{p}{1-p})^{d_{ml}} g_{lo}} > 0
$$

let $\alpha = \frac{1-p}{p}$ then the above can be rewritten as

$$
(d_{no} - d_{nj}) \log \alpha + \log \frac{g_{jo}}{g_{oo}} + \log \frac{g_{oj}}{g_{oo}} - \log \frac{\sum_l \alpha^{-d_{ml}} g_{lj}}{\sum_l \alpha^{-d_{ml}} g_{lo}} > 0
$$

$$
d_{no} - d_{nj} > \frac{1}{\log \alpha} \left(\log \frac{g_{oo}}{g_{jo}} + \log \frac{g_{oo}}{g_{oj}} + \log \frac{\sum_l \alpha^{-d_{ml}} g_{lj}}{\sum_l \alpha^{-d_{ml}} g_{lo}} \right) > 0
$$

The left hand side is maximized when $j = n(d_{nj} = 0)$. In which case an error occurs if

$$d_{no} > \frac{1}{\log \alpha}\left(\log \frac{g_{oo}}{g_{jo}} + \log \frac{g_{oo}}{g_{oj}} + \log \frac{\sum_l \alpha^{-d_{ml}} g_{lj}}{\sum_l \alpha^{-d_{ml}} g_{lo}}\right)$$

If we pick n such that $d_{no} = \min_k\{d_{ko}\}$ then the quantity on the right is the number of bit errors that can be corrected by this system in a span of 2W bits. Examining the RHS we see that this quantity is zero when the conditional probabilities (g_{lk}) are all the same, i.e., the channel input is iid. This validates both the idea that when the source coder removes all redundancy it should not be considered for channel error correction, and the thesis proposed here that redundancy in the source coder output can be used to correct channel errors. The type of redundancy necessary is also evident from the inequality. We wish to increase the variability of g_{lk}. To state this more formally our objective is to minimize $H(y_n|y_{n-1})/M$ where M is the size of the alphabet. The next step, of course, is to examine ways designing (joint) source (channel) coders which contain this type of redundancy. Before we look into that there is one more interesting observation that can be made. α is a decreasing function of p, thus $1/\log \alpha$ is an increasing function of p. This means that for a given source (/channel) coder, an increase in the probability of error in the channel will increase the number of bit errors that can be corrected. Because the number of bit errors also increases, this results in a flattening out of the performance curve. This behaviour was observed in [9].

1.3 Example

A more detailed version of the transmitter side is shown below.

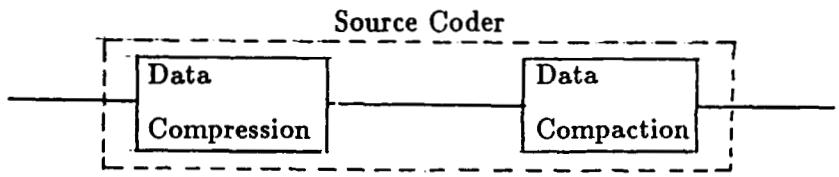

Figure 1.4: A Source Coder

The data compression block consists of source coding algorithms which are not information preserving such as DPCM, Transform Coding, etc. The data compaction block consists of information preserving or noiseless coding algorithms such as Huffman coding or runlength coding. The information preserving compression algorithms are especially vulnerable to channel error as

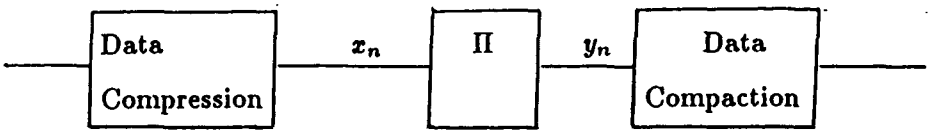

Figure 1.5: Proposed Joint Source/Channel Coder

the error may cause timing and synchronization errors which may propogate for extremely long periods. To be able to correct errors by the technique presented previously, we need to increase the desired type of redundancy. To this end we modify our source coder as follows.

The objective of the block Π is to generate an output sequence y_n, such that $H(y_n|y_{n-1})/M$ is minimum. A simple mapping which does that is as follows. Let the input to Π be selected from the alphabet

$$A = \{a_o, a_1, \ldots, a_{N-1}\}$$

Then let the output alphabet be

$$S = \{s_o, s_1, \ldots, s_{N^2-1}\}$$

Note that while the input alphabet is of size N, the output alphabet is of size N^2. The input/output mapping is given as

$$x_n = a_i, x_{n-1} = a_j \Longrightarrow y_n = s_{jN+i}$$

While we still have to show mathematically that this results in decreasing of $H(y_n|y_{n-1})/M$, we can see the effect. If we look at all pairs $(y_n = s_i, y_{n-1} = s_j)$ we can see that certain pairs are disallowed because of the mapping. For example, the $N^2 - N$ pairs of the form $(y_n = s_o, y_{n-1} = s_j)$ where $j \neq kN(k = 0, 1, \ldots, N-1)$ are disallowed by the mapping. Thus $P[y_n = s_o|y_{n-1} = s_j] = 0$ for $j \neq kNk = 0, 1, \ldots, N-1$. For pairs that are allowed $P[y_n|y_{n-1}] = P[x_n|x_{n-1}, x_{n-2}]$. All this together with the fact that M is now equal to N^2 instead of N means that we have in some sense achieved our objective. Another way of looking at this is that because certain sequences are disallowed, errors which cause such sequences to be generated will be detected and perhaps corrected.

While the mapping Π does seem to increase our error correcting capability what does it cost in terms of additional rate? This is easy to answer if we assume that the data compaction scheme coding rate is equal to the entropy of its input. In the first case the rate is simply $H(x_n)$. In the

second case, as we are actually coding the pairs (x_n, x_{n-1}), the rate is $H(x_n, x_{n-1})$. We can write $H(x_n, x_{n-1})$ as

$$H(x_n, x_{n-1}) = H(x_n) + H(x_n|x_{n-1})$$

Thus the additional cost for this error correcting capability is $H(x_n|x_{n-1})$. To minimize the cost we have to minimize $H(x_n|x_{n-1})$. This is very nice because $H(y_n|y_{n-1})/M$ seems to be a motonic function of $H(x_n|x_{n-1})$. Thus we have identified an objective in the design of the data compression scheme: Minimize $H(x_n|x_{n-1})$.

This system was utilized with an image coding system. The data compression scheme was a 2 bit DPCM system. The source was a 256 x 256 image. End of line resynchronization was assumed. Some preliminary results are shown in Figure 1.6.

We can see from the figure that there is a substantial improvement in performance at low error rates. There is, however, some degradation at higher error rates. The reason for this is that the decoding scheme described above has not completely been implemented. As soon as this is done, we expect a flattening out of the performance curve.

1.4 Continuing Effort

In the next six months we plan to further refine our error correction scheme. This scheme will then be used in conjunction with the algorithms developed for coding of the gamma ray information (section 2).

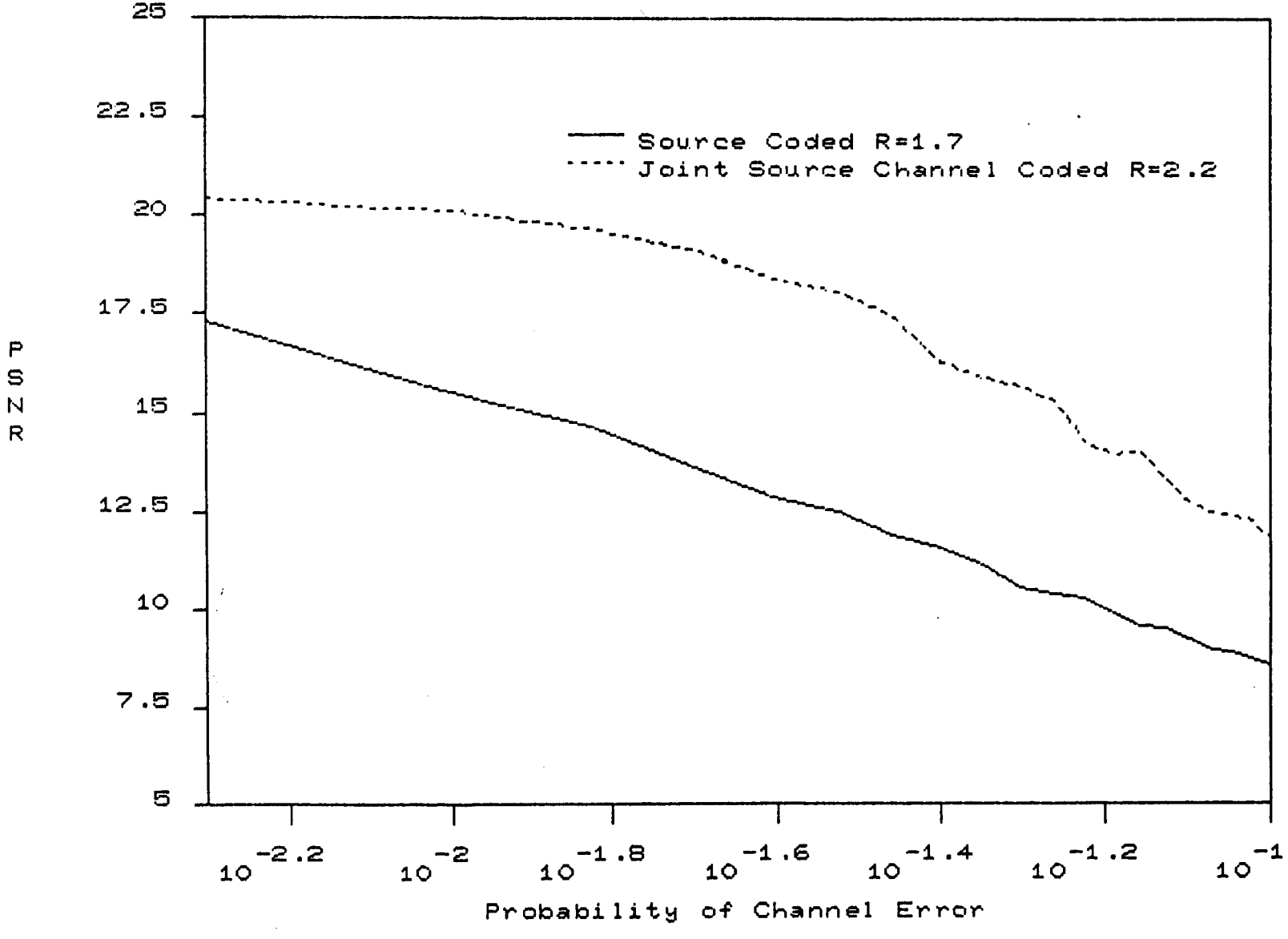

Figure 6: Performance comparison between standard source coding and joint source/channel coding.

1.5 References

1 C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Tech J.*, Vol. 27, pp. 623–656, 1948.

2 J. W. Modestino, D. G. Daut, and A. L. Vickers, "Combined Source-Channel Coding of Images Using the Block Cosine Transform," Vol. COM-29, pp. 1262–1274, September 1981.

3 D. Comstock and J. D. Gibson, "Hamming Coding of DCT Compressed Images Over Noisy Channels," *IEEE Trans. Commun.*, Vol. COM-32, pp. 856–861, July 1984.

4 K-Y. Change and R. W. Donaldson, "Analysis, Optimization, and Sensitivity Study of Differential PCM Systems Operating on Noisy Communication Channels," *IEEE Trans. Commun.*, Vol. COM-20, pp. 338–350, June 1972.

5 D. J. Goodman and C. E. Sundberg, "Transmission Errors and Forward Error Correction in Embedded Differential Pulse Code Modulation," *Bell Syst. Tech. J.*, Vol. 62, pp. 2735–2764, November 1983.

6 A. J. Kurtenbach and P. A. Wintz, "Quantizing for Noisy Channels," *IEEE Trans. Commun. Technol.*, Vol. COM-17, pp. 291–302, April 1969.

7 N. Farvardin and V. Vaishampayam, "Optimal Quantizer Design for Noisy Channels: An Approach to Combined Source-Channel Coding," to appear in IEEE Trans. Inform Theory.

8 K. Sayood, "Analysis of Differential DCM," *Proc.* Nineteenth Arilomas Conference on Circuit, Systems, and Computers, Nov. 1985, pp. 218–222.

9 K. Sayood and J. C. Borkenhagen, "Design of Source Coders and Joint Source/Channel Coders for Noisy Channels," submitted to IEEE Transactions on Communications.

10 R. Steele, D. J. Goodman, and C. A. McGonegal, "A Difference Detection and Correction Scheme for Combatting DPCM Transmission Errors," *IEEE Trans. Commun.*, Vol. COM-27, pp. 252–255, January 1979.

11 "Partial Correction of Transmission Errors in DPCM without recourse to Error Correction Coding," *Elec. Lett.*, Vol. 13, pp. 351–353, June 1977.

12 R. C. Reininger and J. D. Gibson, "Soft Decision Demodulation and Transform Coding of Images," *IEEE Trans. Commun.*, Vol. COM-31, pp. 572–577, April 1983.

Section 2

Efficient Coding and Transmission of Gamma Ray Information from the Mars Orbiter

2.1 Problem Statement

The output of a Gamma Ray detector is quantized using a 14 bit A/D. The number of each of the 16,384 output levels occus in a 30 second interval is obtained. The contents of the 16,384 "bins" are transmitted using a transmission rate of 600 bits per second. This means that the contents of the bins have to be noiselessly encoded using 18,000 bits.

2.2 The Rice Algorithm

The proposed coding algorithm is actually a collection of highly efficient noiseless coding techniques developed by R. F. Rice at the Jet Propulsion Laboratories [1]. The various techniques are used adaptively depending on the changing characteristics of the data. Rice has shown that by adaptively selecting the technique best suited to the data, performance close to the entropy of the source can be obtained for memoryless sources. He shows this to be true for a wide range of entropies. For the range of entropies of interest the set of techniques called the Basic compressor is most appropriate. In the following paragraphs we give a brief description of these techniques. Much more detailed expositions can be found in [1]-[4].

Before any of the techniques due to Rice can be invoked, the data has to be preprocessed to remove correlation. The preprocessed data has to be relabeled into the set of non-negative intergers. The correlation removal operation suggested by Rice [1] is a simple differencing step.

Before detailing the techniques that comprise the Basic compressor some definitions are in order.

Fundamental Sequence: The code operator fs[] is defined by

$$fs[i] = \overbrace{00\ldots01}^{i \text{ zeros}}$$

Let x be a sequence of nonnegative integers

$$\mathbf{x} = x_1 x_2 x_3 \ldots x_J$$

Then the Fundamental Sequence corresponding to $\mathbf{x}$, $FS[\mathbf{x}]$, is given as

$$FS[\mathbf{x}] = fs[x_1] * fs[x_2] * fsx_3 * \ldots * fs[x_J]$$

where $*$ denotes concatenation. As an example take the sequence $\mathbf{x} = 1302$, then $FS[\mathbf{x}] = 010011001$.

Sequence Extension: Let $\mathbf{y}$ be any J sample sequence, then an extended sequence y^e is formed by terminating $\mathbf{y}$ with enough zeros to make the resulting sequence a multiple of e. The e^{th} extension of $\mathbf{y}$ is simply the grouping of $\mathbf{y}^e$ into e-tuples. Suppose $\mathbf{y} = 1101101$ and $e = 3$, then $y^e = 110110100$ and the third extension y^e is $y^e = (110) * (110) * (100)$.

Complementation: Given any binary sequence $\mathbf{x}$, the sequence $\overline{\mathbf{x}} = \text{COMP } [\mathbf{x}]$ is simply the bitwise complement of $\mathbf{x}$.

Coding a Sequence: Given a binary sequence $\mathbf{x}$, the coded version of the sequence $C\mathbf{x}$ is simply the Huffman coded e^{th} extension of $\mathbf{x}$. Thus if e is 3 then $C\mathbf{x}$ is the sequence obtained by coding the 3-tuples of the 3^{rd} extension using a Huffman code designed for an eight letter alphabet.

With these definitions we can now proceed to define the four operators which make up the basic compressor. These are denoted by the symbols $\psi_0, \psi_1, \psi_2, \psi_3$. For a sequence of non negative integers x, the four operators are defined as follows:

1. $\psi_0[\mathbf{x}] = C\overline{FS}[\mathbf{x}]$, i.e., $\psi_0[\mathbf{x}]$ is the coded e^{th} extension of the complement of the fundamental sequence corresponding to $\mathbf{x}$.

2. $\psi_1[\mathbf{x}] = FS[x]$, i.e., $\psi_1[x]$ is the fundamental sequence corresponding to x.

3. $\psi_2[\mathbf{x}] = CFS[\mathbf{x}]$, i.e., $\psi_2[\mathbf{x}]$ is the coded e^{th} extension of the fundamental sequence corresponding to $\mathbf{x}$.

4. $\psi_3[\mathbf{x}]$ is simply the binary representative of $\mathbf{x}$.

Some additional overhead must be tacked on to the ψ_1 and ψ_2 operators. The additional bits record the number of zeros added during the extension process. This information is necessary at the decoder. While this is not mentioned in [1] we found it necessary in our simulations.

The Basic compressor functions as follows: The input is partitioned into blocks. Rice suggests a block size of 16. We found this to be a good choice and have used it in our simulations. Each block is then coded using the "best" operator. The coded sequence is transmitted along with a two bit lable (ID) denoting the operator used. The decision as to which operator is to be used can be made in one of two ways. The first way is to actually code the block using the four operators then select the one which uses the fewest bitrs. The second way is a decision rule proposed by Rice. The decision rule functions as follows. Let x be a sequence of J non negative integers. The length of the fundamental sequence F is

$$F = J + \sum_{i=1}^{J} x_i$$

Four functions $\gamma_0, \gamma_1, \gamma_2, \gamma_3$ corresponding to the four operators can be defined as

$\gamma_0 = \lceil F/3 \rceil + 2(\mathrm{F} - J)$

$\gamma_1 = F$

$\gamma_2 = \lceil F/3 \rceil + 2J$

$\gamma_3 =$ constant

The adaptive algorithm then selects the coding operator corresponding to the minimum γ_i.

2.3 Simulation Result

The Basic Compressor algorithm was simulated on a VAX 11/785. The data coded by the Basic Compressor was generated at the Goddard Space Flight Center by Ms. M. Mingarelli-Armbruster. Table 2.1 shows the coding rate for twenty intervals of thirty seconds each.

The average bit rate required to transmit all twenty intervals is 718.6 bits per second. This is considerably higher than the original goal of 600 bits per second. However, if we compare these results to the results in [1], we find that the two results are in reasonable close agreement. To do the comparison we have to obtain the entropy of the difference original and the number of bits/sample used by the Rice algorithm. In the case of this simulation the entropy is .97 bits while the number of bits/sample is 1.3. Thus there is a difference of .33 bits. This is only slightly higher than the difference of .28 bits obtained by Rice in [1].

Table 2.1: Rates for the Rice Algorithm (Target: 18000 bits)

Interval #	Total # of bits used	Required rate (bps)
1	21647	721.6
2	21385	712.8
3	21530	717.7
4	21562	718.7
5	21666	722.2
6	21424	714.1
7	21841	728.0
8	21630	721.0
9	21719	723.9
10	21568	718.9
11	21308	710.3
12	21509	716.9
13	21633	721.1
14	21822	727.4
15	21296	709.8
16	21701	723.4
17	21058	701.9
18	21312	710.4
19	21713	723.8
20	21888	729.6

More disturbing than the fact of a higher than expected rate, however, is the behavior of the algorithm in the presence of channel noise. Table 2.2 are from a hundred different runs. It was assumed that the receiver was resynchronized every 30 seconds.

Table 2.2: Effect of Noise on the Rice Algorithm

Probability of Error	Mean Squared Error	Mean Absolute Error	Number of Errors
10^{-3}	437.8	15.8	15,689
10^{-4}	39.9	3.3	10,228
10^{-5}	4.1	.4	1,773

The number of errors column shows how many of the 16,394 values were received incorrectly. As can be seen from the table at a probability of error of 10^{-3} effectively all the received values are in error; at a probability of error of 10^{-4} about two thirds of the received symbols are in error; and, even at probability of error of 10^{-5} more than 10% of the received symbols are in error.

There are two main reasons for this lack of robustness of the system. Firstly, by its very nature, an adaptive system is vulnerable to noise, as an error at the receiver may cause it to mistake the

coding scheme used at the transmitter. Secondly the differencing operation necessary to create the uncorrelated sequence required by the Rice algorithm, also creates infinite memory at the receiver. This means that once an error has occurred, it will propogate until resynchronization occurs. In these situations we have assumed that resynchronization occurs at the end of each thirty second interval. If this is not true the effect of errors may be even more disastrous.

Because of our concerns with the Rice algorithm under the current conditions we investigated the possiblity of developing an alternative algorithm.

2.4 Possible Alternate Algorithm

The first step in our alternative algorithm is also a difference, only it is a leaky differencer. The difference signal is obtained as

$$d(n) = x(n) - \lfloor \frac{3}{4} x(n-1) \rfloor$$

The leakage in the differencer causes error effects to die out in time. The difference signal is encoded using a sixteen symbol modified runlength code. These sixteen symbols can then be encoded using either a four bit fixed length code or a variable rate Huffman code. The results of using this algorithm to encode the same twenty intervals is shown in Table 2.3 where VR stands for variable rate code and FR stands for fixed rate code.

If we use the fixed rate code, the bit rate required is 594 bits per second which is below our target rate. The use of the variable rate code would require a channel rate of 522 bits per second. As both the fixed and variable rate codes will meet the target rate and as the fixed rate code is both robust and simple to use, we elected to go with the fixed rate code.

Table 2.3: Coding Rates for Alternative Algorithm (Target: 18000 bits)

Interval Number	# of bits used(FR)	# of bits used (VR)
1	17382	15733
2	17528	15345
3	17784	15520
4	17840	15691
5	18144	15883
6	17504	15457
7	18048	15882
8	18096	15907
9	18132	15843
10	18096	15695
11	17604	15438
12	17728	15580
13	17780	15581
14	18016	15913
15	17564	15361
16	17956	15872
17	17296	15139
18	17688	15449
19	18160	16033
20	18292	16125

The effect of noise on this system is tabulated in Table 2.4.

Table 2.4: Effect of Noise on the Alternative Algorithm

Probability of Error	Mean Squared Error	Mean Absolute Error	Number of Errors
10^{-3}	1.336	0.329	3161
10^{-4}	0.426	0.111	1384
10^{-5}	0.109	0.035	459

Comparing table 2.4 to table 2.2, we see an improvement by an order of magnitude in the number of errors and several orders of magnitude in the mean squared error. Especially striking is the mean squared error at a probability of error of 10^{-3}. At this probability of error use of the Rice algorithm results in an error of **437.8** while the use of the alternative algorithm results in an error of **1.3**!

While this algorithm is still in its preliminary stages and requires considerable testing, we feel that these results make it attractive enough to pursue in further detail.

2.5 Continuing Effort

We have spent the first six months of this project evaluating the Rice algorithm and developing a possible alternative algorithm. In the next six months we will develop error correction algorithms for both these algorithms, at which time we will be better able to evaluate both algorithms. We also plan to look at ways of combining the two algorithms for improved performance.

2.6 References

1 R. F. Rice, "Some Practical Universal Noiseless Coding Techniques," Jet Propulsion Laboratory, California Institute of Technology, JPL Publication 79–22.

2 R. F. Rice, "Practical Universal Noiselss Coding," 1979 SPIE Symposium, Vol. 207, San Diego, CA, August 1979, pp. 247–257.

3 R. F. Rice and A. P. Schlutsmeyer, "Software for Universal Noiseless Encoding," Proceedings of the 1981 International Conference on Communication.

4 R. F. Rice and Ju-Ji lee, "Some Practical Universal Noiselss Coding Techniques, Part II," Jet Propulsion Laboratory, California Institute of Technology, JPL Publication 83–17.

Section 3

High Fidelity Low Rate Coding of Images

Let **F** be an $N \times N$ image segment. **F** can be transform coded into a 2-dimensional representation

$$\mathbf{C} = \mathbf{T}(\mathbf{F}).$$

If the transform is linear, **C** can be represented as a double sum

$$C_{ij} = \sum_{k,l=1}^{N} T_{ijkl} F_{kl}$$

Here the i and k subscripts represent x-transform coefficients in the image and transform spaces. Likewise, j and l represent y-transform coefficents. This 2-dimensional transform can be converted into a scalar transform by stacking the image and transform matrices,

$$c_p = \sum_{m=1}^{N^2} t_{pm} f_m \tag{3.1}$$

where **F** has been stacked into a vector **f** of size $N^2 \times 1$, **T** has been stacked into **t** a matrix of size $N^2 \times N^2$, and **C** is a $N^2 \times 1$ vector of the transform coefficients. Eq. (3.1) is attained using the following stacking operation on the i, j, k, l indices:

$$m = N(l-1) + k$$

$$p = N(i-1) + j.$$

The transform vector **c** must be quantized if data compression is to be attained,

$$\hat{c}_p = Q_p(\alpha_p c_p)$$

where the α_p are scaling factors to match the variance of the c_p to that of the quantizer codebook. In general, a different codebook can be used for each of the N^2 coefficients of the transform.

The coding method used here keeps only the 4 lowest frequency coefficients of the transform (i.e., DCT, Hadamard) so there are only 4 $\hat{c}_p$ elements. Thus, t is a $4 \times N^2$ matrix. To map the non-zeroed elements of c into $\hat{\mathbf{c}}$ the following quantizer mappings are needed:

$$\hat{c}_1 = Q_1(\alpha_1 c_1) \text{ dc coefficient}$$
$$\hat{c}_2 = Q_2(\alpha_2 c_2)$$
$$\hat{c}_3 = Q_2(\alpha_3 c_3)$$
$$\hat{c}_4 = Q_2(\alpha_4 c_4)$$

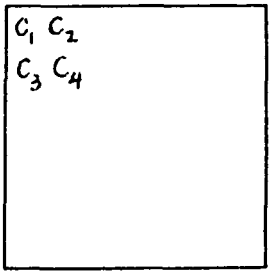

Figure 3.1: Only 4 transform coefficients are used for each image segment.

The non-dc elements of $\hat{\mathbf{c}}$ are distributed in a similar fashion and can be quantized with the same quantizer. The dc coefficient is distributed differently than the other coefficients and requires a different quantizer to attain the best results. For the work done here Q_1 is coded to 8 bits and Q_2 is coded to 5 bits.

The image can be recovered from the 4 transform coefficients by solving (3.1),

$$\hat{\mathbf{f}} = (\mathbf{t}^T\mathbf{t})^{-1}\mathbf{t}^T\hat{\mathbf{c}}$$

If the rows of t are ortho-normal this reduces to $\hat{\mathbf{f}} = \mathbf{t}^T\hat{\mathbf{c}}$.

Let us define two distortion measures to rate the performance of the above 4 coefficient transform method:

1) $d_1 = \max_i |f_i - \hat{f}_i|$

2) $d_2 = \frac{1}{N^2-1}\sum_{i=1}^{N^2}(f_i - \hat{f}_i)^2 = E\{(f-\hat{f})^T(f-\hat{f})\}$

Method 1) indicates the largest absolute error between the original and reconstructed images.

Method 2) is the common MSE measure where the signal-to-noise ratio is defined to be

$$SNR(dB) = 20\log_{10}\frac{d_2^2(N^2-1)}{\mathbf{f}^T\mathbf{f}}$$

and the peak signal-to-noise ratio is

$$PSNR(dB) = 20\log_{10}\frac{d_2^2(N^2-1)}{\mathbf{w}^T\mathbf{w}}$$

where $\mathbf{w}$ is a $N^2 \times 1$ vector whose elements are set to the white level of the image, i.e., 255 for an 8-bit image.

Notice that for method 1) $|f_i - \hat{f}_i| \leq d_1$ for all pixels in $\mathbf{f}$ but it can be possible that $d_2 << d_1$ if the transform coefficients represent the original image well except for a few pixels where the distortion can be large. This is true since d_2 is a measure of average distortion over the entire image segment. So a good d_2 value will cause one to think that the image is coded with a good match but, in fact, there may be areas of very large local distortion in the image.

To overcome the shortcoming of the d_2 distortion method, an image segment is coded and the d_1 distortion is measured, and then, depending whether or not d_1 is less than some distortion threshold, t, the segment may be subdivided into 4 $\frac{n}{2} \times \frac{n}{2}$ segments, $\mathbf{F}_i, i = 1,2,3,4$, and each segment is again transform coded and checked against the distortion threshold. If a segment fails the threshold test it is again subdivided until the threshold level is meet or a minimum block size for an image segment is attained.

Notice, that if the minumum block size is 2×2, the 4 coefficients of the above method will code the image segment exactly, to within the quantization resolution of the coder.

Method

The method for image coding by threshold detection as decribed above is outlined below:

1. Select an image segment $\mathbf{F}$.

2. Code $\mathbf{F}$ by $\hat{\mathbf{F}}$.

3. If $d_1(\mathbf{F} - \hat{\mathbf{F}}) < t$, $\hat{\mathbf{F}}$ is an adequate representation of $\mathbf{F}$.

   If $d_1(\mathbf{F} - \hat{\mathbf{F}}) < t$, divide $\mathbf{F}$ into 4 $\frac{n}{2} \times \frac{n}{2}$ sub-segments $\mathbf{F}_i, i = 1,2,3,4$, and go to 2) for each segment until $d_1 < t$ or the minimum block size is attained. If the minimum block size is attained code $\mathbf{F}_i$ by $\hat{\mathbf{F}}_i$.

Example

Consider the coding of a 32 × 32 image segment. Let the maximum coding block size be 16 × 16 and the smallest block size be 4 × 4. Let the original image segment be coded with 8 bits per pixel.

If the 32 × 32 segment is coded with four 16 × 16 blocks the coding rate for each such block is

$$\frac{8+5+5+5}{16^2} = .090 \text{ bits/pixel}$$

Let the threshold level for the d_1 distortion measure be $t = 5$. This means that if the coded 16 × 16 blocks have a d_1 level $< t$ the maximum distortion at any given pixel in 2 bits (4 gray scale levels). For the example at hand let the d_1 levels for the four 16 × 16 blocks be as shown in Figure 3.2.

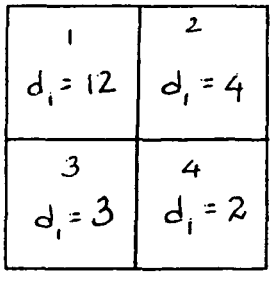

Figure 3.2: Image segment **F** coded in 16 × 16 blocks.

In this case, $\mathbf{F}_2, \mathbf{F}_3$, and $\mathbf{F}_4$ all meet the d_1 distortion threshold so they are adequately coded but, $\mathbf{F}_1$ does not meet the distortion threshold, so it must be divided into four 8 × 8 segments and coded again. Figure 3.3 represents the d_1 distortion profile of subsegment $\mathbf{F}_1$.

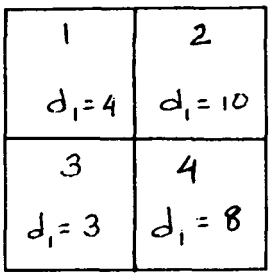

Figure 3.3: Subsegment F_1 coded in 8 × 8 blocks.

In this case, segments 1 and 3 meet the distortion threshold and are adequately coded but, segments 2 and 4 must be divided into 4 × 4 blocks and recoded. Before doing this, let us calculate the coding rate for the original image up to this point into the process. The remaining 16 × 16 block that has been divided into four 8 × 8 blocks require 23 bits for each block. So the number of coding bits for the entire 32 × 32 image segment is

$$3(28) + 4(23) = 161 \text{ bits}$$

and if the smallest block size was 8 × 8 then the coding rate for this total 32 × 32 image is

$$161/32^2 = .157 \text{ bits/pixel.}$$

Since the minimum block size is 4 × 4 and two of the 8 × 8 blocks do not meet the d_1 distortion threshold, the segment F must be divided for coding as shown in Figure 3.4.

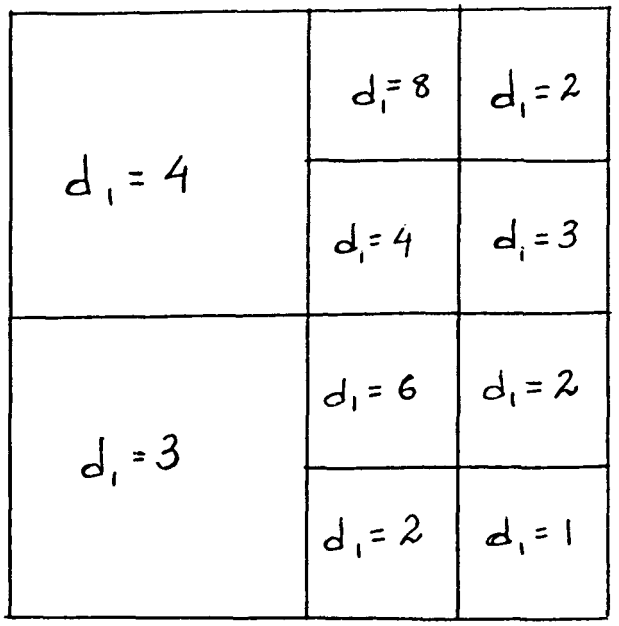

Figure 3.4: The final segmenting of segment F_1

Now all of the blocks of the original image meet the d_1 distortion threshold except for two 4×4 blocks whose d_1 levels are 8 and 6. (If the 4×4 blocks could be divided into 2×2 blocks then this situation could be improved, but, for this example, the 4×4 blocks are the smallest segments to be coded.) The final number of bits to code the original 32×32 image segment is

3(23) for 3 – 16×16 blocks;

2(23) for 2 – 8×8 blocks;

8(23) for 8 – 4×4 blocks; for

for a total of 299 bits. So the coding rate is $299/32^2 = .292$ bits/pixel for a data compression ratio of

$$8/.292 = 27.4.$$

When the receiver gets these 299 bits of coefficient data it must know how to apply the coefficients to reconstruct the subsegments of the 32×32 image segment correctly. To do this, a small amount of side information must be transmitted.

Let four bits of side information be transmitted for each image segment that can be subdivided. These bits will be set to 1 if the corresponding subsegment is to be divided. The bit is set to 0 if is not to be divided. Since only segment 1 of the 32×32 image is to be divided, the four bits of side information are coded 1000. Since this 16×16 segment has two 8×8 subsegments to be divided again, namely blocks 2 and 4, the four bits of side information for this subsegment are 0101. Thus,

the full string of side information is 1000,0101, for a total coding bit rate of

$$\frac{299+8}{32^2} = .300 \text{ bits/pixel}$$

and a final data compression ratio of 26.7.

When the receiver has the side information string it will know that the three 0's of the first block say that the corresponding 16 × 16 segments are to be coded as single blocks. The first bit is a 1 so the receiver will need to look at the second side block to see how to subdivide its 16 × 16 block. In the sideblock the two 0's say that blocks 2 and 3 will be coded as 8 × 8 segments and blocks 1 and 4 must be divided into 4 × 4 blocks. So the 8 bits of side information tell the receiver how it must subdivide the original 32 × 32 image segment as shown in Figure 3.5.

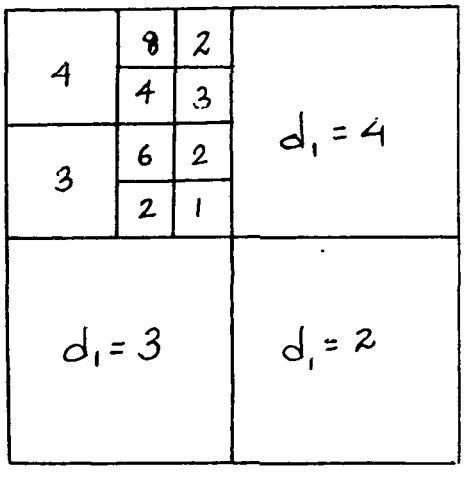

Figure 3.5: The subdivided 32 × 32 image segment with d_1 distortions and required side information.