JPL Publication 94-16

System Considerations for Efficient Communication and Storage of MSTI Image Data

Robert F. Rice

October 1994

Prepared for

Ballistic Missile
Defense Organization
Through an agreement with

National Aeronautics and
Space Administration

by

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Ballistic Missile Defense Organization (BMDO) through an agreement with the National Aeronautics and Space Administration.

ABSTRACT

The Ballistic Missile Defense Organization has been developing the capability to evaluate one or more high-rate sensor/hardware combinations by incorporating them as payloads on a series of Miniature Seeker Technology Insertion (MSTI) flights. This publication represents the final report of a 1993 study to analyze the potential impact of data compression and of related communication system technologies on post-MSTI 3 flights.

Lossless compression is considered alone and in conjunction with various spatial editing modes. Additionally, JPEG and Fractal algorithms are examined in order to bound the potential gains from the use of lossy compression. But lossless compression is clearly shown to better fit the goals of the MSTI investigations. Lossless compression factors of between 2:1 and 6:1 would provide significant benefits to both on-board mass memory and the downlink. For on-board mass memory, the savings could range from \$5 million to \$9 million. Such benefits should be possible by direct application of recently developed NASA VLSI microcircuits.

It is shown that further downlink enhancements of 2:1 to 3:1 should be feasible through practical modifications to the existing modulation system and incorporation of Reed-Solomon channel coding. The latter enhancement could also be achieved by applying recently developed VLSI microcircuits.

ACKNOWLEDGMENTS

The system considerations provided here are based on assumptions for data/ sensor and communication system parameters obtained from Robert Denard and Marc Coleman of Phillips Laboratory and Ray Rew of Spectrum Astro.

TABLE OF CONTENTS

Figures

Tables

I. BACKGROUND/PROBLEM DEFINITION

The Ballistic Missile Defense Organization (BMDO) has been developing the capability to evaluate one or more high-rate sensor/hardware combinations by incorporating them as payloads on a series of Miniature Seeker Technology Insertion (MSTI) flights. This publication represents the final report of a 1993 study to analyze the potential impact of data compression and of related communication system technologies on post-MSTI 3 flights.

For the purposes of discussion here, it is assumed that the principal purpose of a MSTI flight is to provide the capability to thoroughly evaluate the characteristics of one or more image sensor/hardware combinations. This includes the ability to determine not only that a system worked, but if it didn't, why? The following discussions will make certain system **baseline** parameter assumptions and then identify how data compression can improve MSTI capability to achieve these evaluation goals and others as well.

BASIC PROBLEM

During the lifetime of a flight there can be expected to be several **Sensor Events** which will nominally last only 10 minutes each. But the data generated during a Sensor Event is at a much higher data rate than can be communicated in real-time. This is the fundamental problem! The data system must be such that for each Sensor Event:

1a) Sensor/hardware operation can be verified.

1b) If operation is flawed, one can determine why.

1c) A full visual demonstration of targeting is possible.

1d) Sample real data from distinct scenarios is provided for

- Future analysis.

- Verification/upgrade of "**K**inetic Energy Weapons **H**ardware **I**n the **L**oop **S**imulator" (KHILS) simulation capability; KHILS is located at Eglin Air Force Base.

1e) Ecological monitoring.

Baseline Parameters

Figure 1 illustrates the key elements of the on-board data acquisition system and uses parameters obtained from meetings with Phillips Laboratory and Spectrum Astro in early 1993.

Multiple sensors generate image data at 30 frames/second (f/s) for a Sensor Event's time period which we will henceforth assume to be **10 minutes**. Nominal frame sizes are a **baseline** 256 picture elements x 256 lines at 8 bits/pixel (b/p), but frame sizes may be as high as 1024 x 1024 x 12. Use of the latter specification will correspond to worst case (maximum) conditions in future discussions.

Each "event" is stored in mass memory for later transmission. A baseline sensor generates approximately 15 Mbits/s, which for a full Sensor Event interval (10 minutes) results in an accumulation of about 1 Gbyte of data or 18,000 frames. However, a single 1024 x 1024 x 12 sensor could generate 28 Gbytes of data for a Sensor Event, and do so at 377 Mbits/s.

What Data is Needed?

Figure 2 focuses on individual frames of image data. On-board tracking can be expected to maintain lock on "something." If this something is the desired "target," the sensor/hardware combination is working. If the tracker veers off from the real target, it is important to be able to discover why by analyzing the data in the many frames generated during the same time period (i.e., those the tracking system would have used when the system failed).

The target itself might typically lie within a 16 x 16 area during proper operation (assuming the baseline 256 x 256 size image).

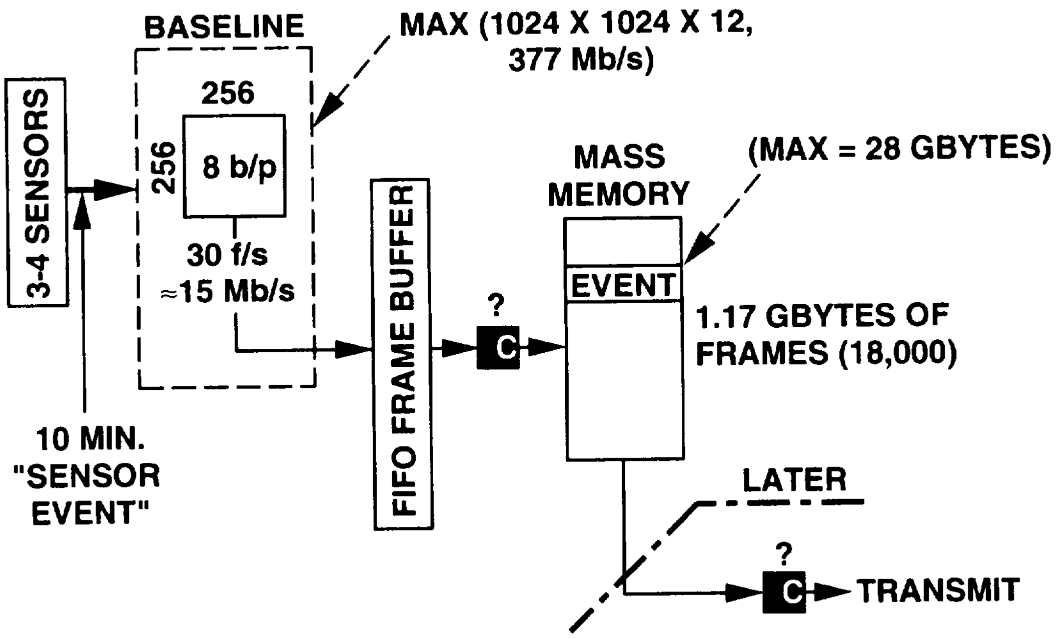


Fig. 1. Key On-board Data System Elements.

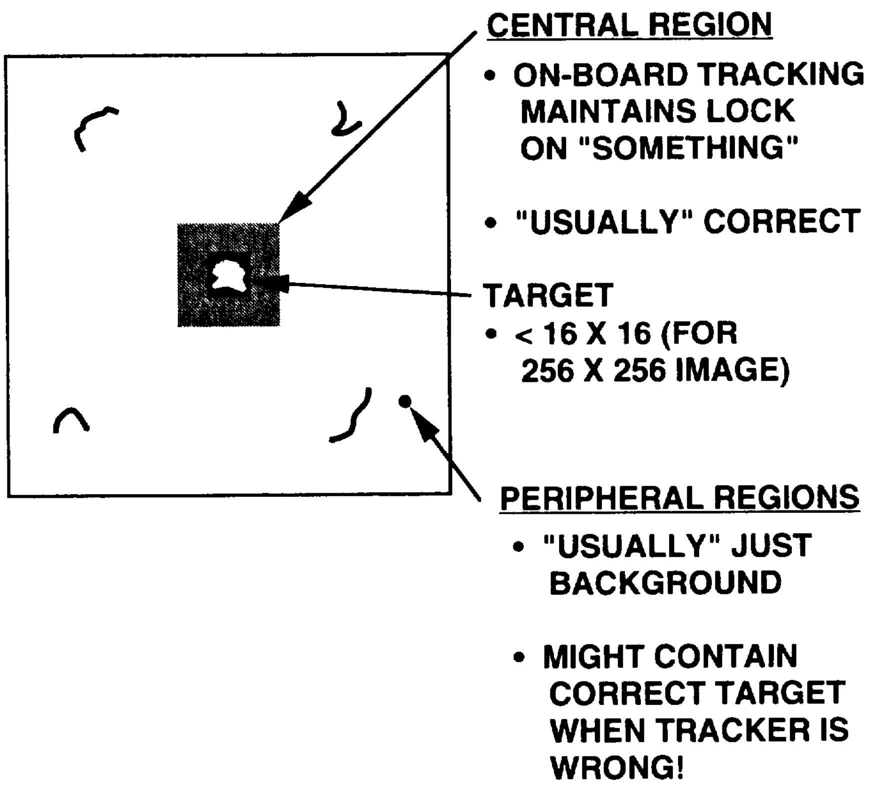

CENTRAL REGION
• ON-BOARD TRACKING MAINTAINS LOCK ON "SOMETHING"
• "USUALLY" CORRECT
TARGET
• < 16 X 16 (FOR 256 X 256 IMAGE)
PERIPHERAL REGIONS
• "USUALLY" JUST BACKGROUND
• MIGHT CONTAIN CORRECT TARGET WHEN TRACKER IS WRONG!

Fig. 2. The Content of Frames

Meeting the Goals in Paragraphs (1a) through (1e)

2a) If we are seeking full verification that some sensor/hardware combination works, only the central area (containing the target) is needed. Probably close to full frame rate is desired for this central area.

2b) If the system fails, then all the data in all the frames in the time period around the failure are needed (at full fidelity).

Since when and if such an event might occur is unknown, **this implies that no data should be discarded or degraded on its way into mass memory.**

2c) To visually demonstrate what a sensor "saw" (e.g., make a movie), a significantly lower frame rate at some possibly lower quality is probably acceptable.

2d) To provide complete data sets, clearly all the data must be recovered without degradation.

2e) To demonstrate sensor capability to support "ecological monitoring" suggests a requirement for gathering sets of **full quality images** for ground processing and study. No data in these sets would be discarded or degraded on their way into mass memory. However, these sets of images should be far smaller than those gathered for the primary MSTI mission. (This is because such images can be acquired at a relatively very low frame rate.) **This requirement should thus have negligible impact on data storage and downlink requirements.**

These are all important observations and will directly impact the approaches to data compression which are treated later.

DATA SYSTEM ELEMENTS AND TERMS

For our purposes at this point, **Data Compression** can be looked at as **any mechanism** for representing a **set of data**. (The sets of data we'll be interested in are various sub-sets of image data generated by a MSTI Sensor Event.) Of course, the goal is to reduce the number of bits required, compared to a Standard Representation (i.e., uncompressed). The factor by which the number of bits is reduced is usually called the Compression Factor (CF). If a Standard Representation is used, CF=1.

When dealing with images, a normalized image rate in bits/pixel (b/p) is often given. This typically reflects the number of bits originally used to quantize each individual pixel (e.g., the 8 b/p assumed for a baseline sensor). A compression rate is then the **average** b/p required by a particular compression scheme.

Rate/Quality. Generally, a compressed data set must be "decompressed" or **reconstructed** back into an **approximation** of the Standard Representation before it can be used. When the reconstruction produces an exact reproduction of the original Standard Representation, the method of compression is said to be "lossless" or "noiseless". Otherwise, when the reconstruction produces an imperfect reproduction of the original Standard Representation, the method of compression is called "lossy".

Figure 3 illustrates the relationship between rate and quality (of reconstruction) relative to the uncompressed image data, when data compression is applied to individual image frames (intraframe compression).

Lossless compression of 2:1 is typically achievable with **intraframe** compression. But the precise CF is data dependent and could be slightly less than 2:1 or as much as 3:1 for MSTI-like data. Of course, the algorithms used can make a difference, too. In particular, using information from adjacent frames (interframe) **could** substantially increase the possible compression factor for lossless compression of some forms of MSTI data. We will call this kind of compression "Frame-to-Frame Lossless" or "F-to-F" lossless for short. In providing estimates of system impact, later discussions will assume an estimated average CF = 6:1 for F-to-F lossless compression.

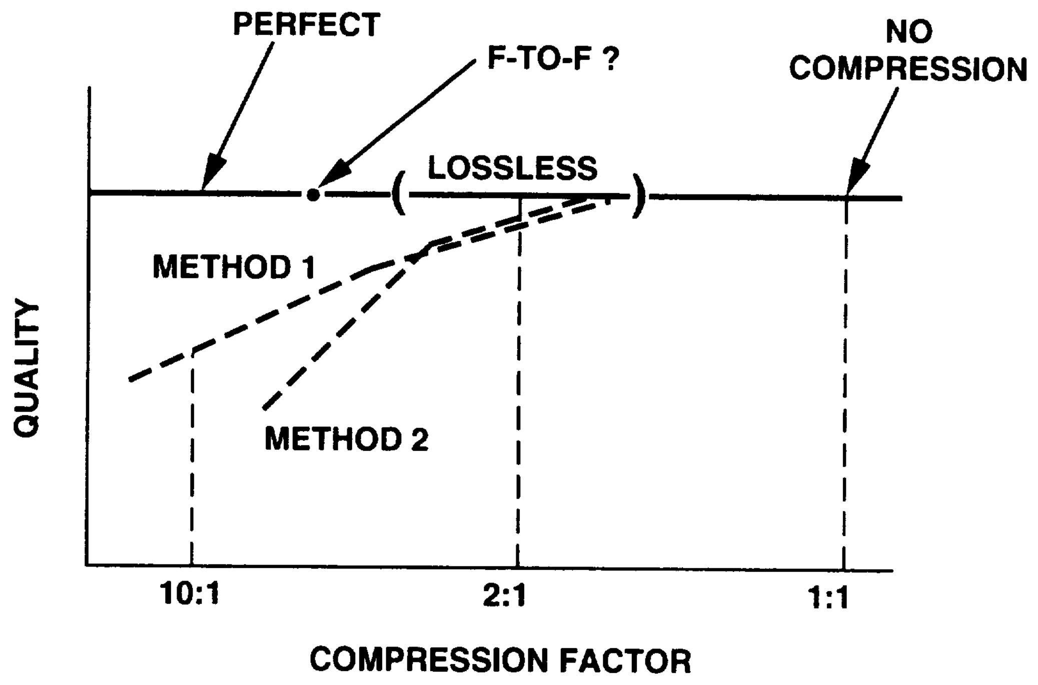


Fig. 3. Rate/Quality Relationships.

If errors are allowed in the reconstruction of some data set, the general rule is that higher compression factors are achievable (as compared to lossless). Again, the methods used can substantially impact the achievable CF (for a specified level of quality). These points are illustrated in Fig. 3.

Rice Algorithms. This report will ultimately demonstrate major advantages of using JPL-developed adaptive lossless compression algorithms for MSTI needs. These have been demonstrated to be superior to any other lossless techniques for imaging and have recently been implemented as high-speed custom micro-circuits. Discussions of Baseline Intraframe lossless compression will assume the application of these chips. Where applicable, F-to-F variations in lossless coding can probably make use of the same chips, with some external hardware. Appendix A provides a discussion of the primary Rice Algorithms, current VLSI implementations and a list of references. Appendix A mentions a JPL chip being developed for the deep-space Cassini Project. This same chip was tested at KHILS for an application to the MSTI 3 downlink.

Lossy Compression. For those situations where imperfect reproduction is acceptable for at least some of the data generated by a Sensor Event, we will investigate the consequences of several approaches. These will include simple spatial/temporal editing modes (which can also make use of lossless coding) as well as the commercial JPEG algorithms and powerful systems under development which use FRACTALS.

END-TO-END BLOCK DIAGRAM

Figure 4 shows how the major end-to-end data system elements and data compression fit together.

As shown, one or more sensors generate high rate image data (see Fig. 2) of which a small subset can be transmitted to the ground in real-time. This small subset cannot satisfy all the goals in paragraphs (1a)-(1e), so further discussion will focus on the non-real-time acquisition and return of "required" data. See paragraphs (2a)-(2e).

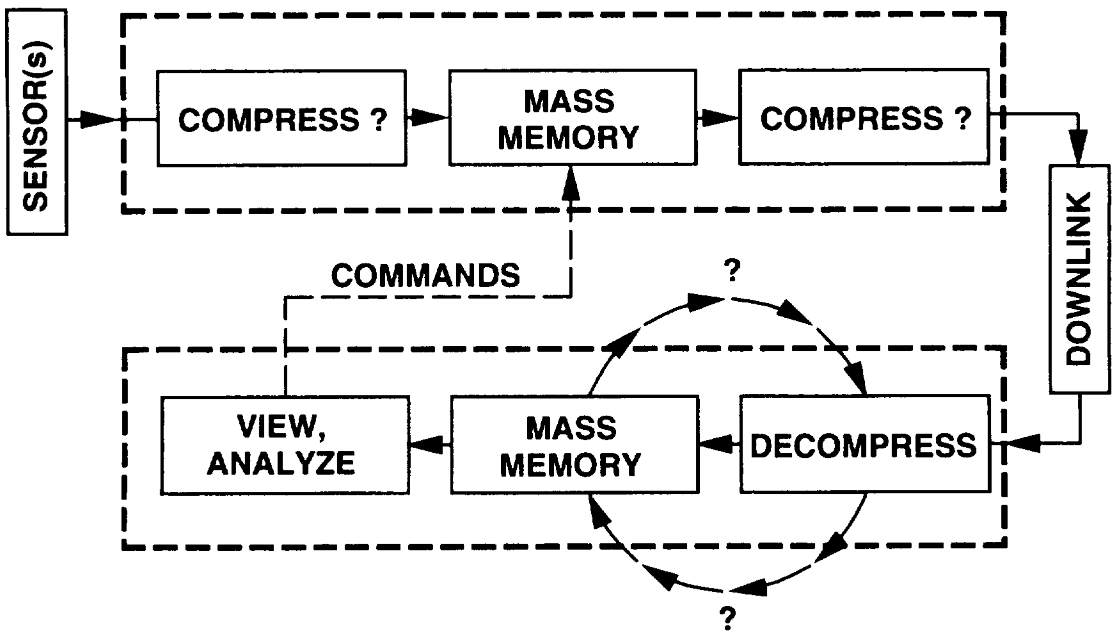


Fig. 4. End-to-End Data System Elements

The high rate image data is placed in a mass storage device and then later downlinked to the ground for processing, display, and analysis. As we'll see, it may be desirable to make this downlink process "interactive" because of the length of time needed to transfer all "desired" data from a Sensor Event. In this case, received data is immediately viewed to determine the content of subsequent transmissions (see the line designated "Commands" in Fig. 4).

Compressor Location On-Board

The on-board data system in Fig. 4 shows two boxes labeled "compress". This is because data compression applies to both locations. For several reasons, not all of them technical, a MSTI 3 demonstration of lossless compression will place the compressor **after** mass memory. However, for post MSTI 3 missions it is not yet clear where it will be possible to locate it.

As noted in paragraphs (2a), (2d) and (2e), goals (1a), (1d) and (1e) cannot be achieved unless all the data is placed into mass memory in a form which can ultimately be reconstructed on the ground **without error**. That is, **if compression is used before storage in mass memory, it should be lossless compression.**

Ultimately, the advantage in applying compression **before storage in mass memory** is:

a) A reduction in memory requirements, or
b) An increase in memory capability for a given memory capacity.

Both of these can translate into significant dollar savings. This potential is quantified in a later section.

Note that the placement of the compressor after mass memory on the MSTI 3 demonstration precludes achieving either of these advantages before MSTI 4.

There are some subtle **implementation consequences** to compressing before storage in mass memory. Later, we will identify lossy compression modes **for data received from mass memory**. These include spatial/temporal formatting techniques (later identified as BGFGs), which make use of lossless coding, JPEG algorithms and FRACTALS. The **JPEG and FRACTAL techniques can only be applied to uncompressed data. This would imply that data which had**

been placed into mass memory compressed (Lossless), would need to be "decompressed" before these other techniques could be used.

Implementing these spatial/temporal formatting techniques using only lossless coding would **not necessarily** require an on-board lossless decompressor, but could require careful attention to the manner in which compressed data was "data based" into memory. This in turn could influence the choice of "compression chip" used (i.e., it could be different than the schedule-mandated choice for MSTI 3).

The Downlink

The advantages of data compression to the downlink process are much like those in the situation for on-board mass memory. The link rate is inadequate, under almost any realistic assumptions. Data compression decreases the time needed to communicate through a data link of a specific capacity. But the potential CF for downlinking is higher than for on-board mass memory because **some situations** may not require lossless reconstruction of all the data. This subject is treated later.

Information provided by Ray Rew of Spectrum Astro in March 1993 led to a proposal for a future practical upgrade of the uncoded 1 Mbit/s MSTI 3 downlink, at a bit error probability of 10^{-6}, to from 2 to 3 Mbits/s with far superior error characteristics. This proposal is repeated as Appendix B and includes discussion on the interaction of compressed data and channel errors.

The Ground Data System

The position of a decompressor relative to mass memory on the ground cannot yet be determined, as illustrated in Fig. 4.

Thus far there has been virtually no discussion of ground concerns. Our assumption has been that decompression could always be done in non-real-time, using software methods. This will certainly be the case for **initial compression experiments**. Thus decompression would initially follow extraction from mass memory (although the decompressed data would probably still reside in some other mass memory form, not shown in Fig. 4). But in the longer run, this area needs greater consideration.

For example, to interactively respond to transmitted data, perhaps altering what is sent next, a decompressor would need to operate as fast as the downlink produces the data. Assuming a 1 Mbit/s downlink, data compressed 2:1 (lossless, intraframe – the baseline) would need to be decompressed into display data at a 2 Mbit/s rate. A software approach might be feasible. But if the compression factor were 50:1, decompressed data would generate 50 Mbits/s of display data. A purely software solution is unlikely.

This is simply one of numerous ground considerations that ultimately influence the viability of various compression approaches discussed in later sections. Ground processing is part of the overall data system and needs to be considered in concert with on-board data system developments.

II. ON-BOARD MASS MEMORY IMPACT

Table 1 illustrates the potential advantages to on-board mass memory capacity requirements based on a number of assumptions, specified below:

1) Sensor Event Parameters:

   Frame Rate = 30 f/s
   Collection Interval = 10 minutes

2) Baseline intraframe lossless compression achieves an average CF of 2:1.

3) Frame-to-Frame (F-to-F) lossless compression achieves an average CF of 6:1.

4) A baseline Sensor Event for one sensor (256 x 256 x 8) requires 1.17 Gbytes of memory.

5) Worst Case (maximum) assumption of an uncompressed Sensor Event for a single sensor (1024 x 1024 x 12) is 28 Gbytes of data.

Then Table 1 shows, for 1 and 4 sensors, the total accumulation for a single Sensor Event, with and without lossless compression.

Compression Value

Consider assigning a value to the application of lossless compression by the following:

Let

$$\theta = \text{Total Cost of Memory if compression is not applied.} \qquad (1)$$

Table 1. Mass Memory Data Accumulation

		ACCUMULATION TOTAL, GBYTES	
	CF	1 SENSOR	4 SENSORS
BASELINE SENSORS	1:1	1.17	4.68
	2:1	0.58	2.34
	6:1	0.19	0.78
MAX. FORMAT SENSORS	1:1	28	112
	2:1	14	56
	6:1	4.7	18.7

The **Compression Value** (CV) is then defined as the difference in the cost of memory, with and without compression:

$$CV = \theta - \frac{\theta}{CF} \tag{2}$$

where θ/CF is the cost of memory needed when compression is used.

Various estimates for the true cost of "flight" mass memory have been put forth. These have ranged between

$$\$100\ K/Gbyte \tag{3}$$

up to

$$\$1\ million/Gbyte \tag{4}$$

Figure 5 illustrates the cost/value (θ and CV) of Eq. (2) for both of these assumptions. Simply interpret the ordinate as either in $100K units or in $1 million units.

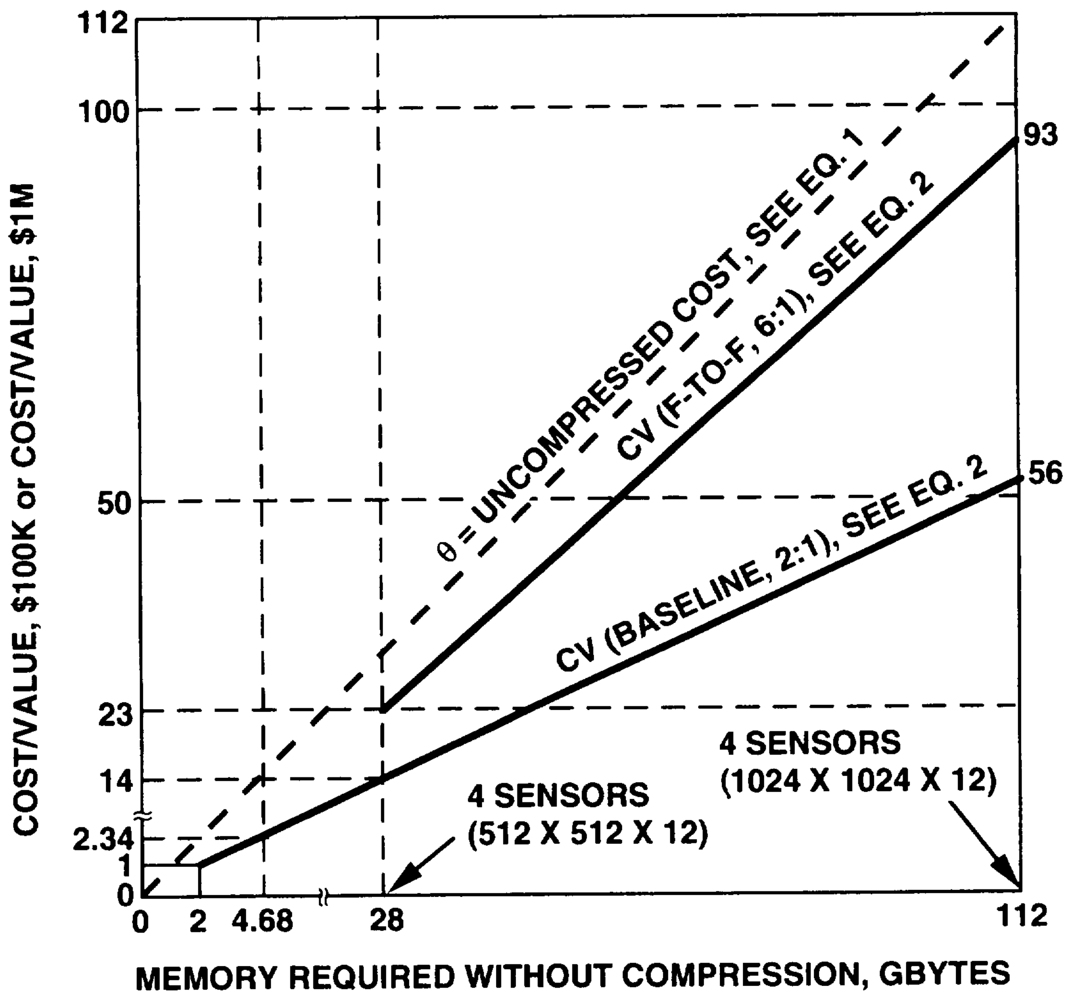


Fig. 5. Mass Memory Compression Value

If we consider the baseline as four sensors with the minimum (256 x 256 x 8) format, 4.68 Gbytes are needed. Assuming a cost of $100K/Gbyte, the uncompressed memory requirement is $468K and the compression value is 468 – 468/2 = $234K assuming a 2:1 compression. But this is $2.34 million at the other end of the scale. Assuming large format sensors, requiring a worst case 112 Gbytes (for four 1024 x 1024 x 12 sensors), the value of compression is a **minimum** of $5.6 million for 2:1 and $9.3 million for 6:1 (F-to-F), assuming the lowest cost of $100K/Gbyte.

If the cost of memory were only double the minimum assumption of $100K/Gbyte, the potential value of compression would go to $11.2 million and $18.6 million respectively for 2:1 and 6:1. This would suggest that performing even the 2:1

compression **before** loading in mass memory should have value far exceeding compression cost.

Collection Interval

Another way to look at the impact of compression and memory is to assess the limitations imposed on the allowed collection interval. Figure 6 shows the impact when the worst-case sensor complement of four 1024 x 1024 x 12 sensors is assumed. Without compression 112 Gbytes of mass memory are required to achieve the "desired" full 10 minute Sensor Event. Only 56 Gbytes are needed if 2:1 compression is to be achieved (baseline lossless compression) and only 18.6 Gbytes if a 6:1 F-to-F capability were achieved. **But without compression**, having "only" 19 Gbytes would limit collection intervals to less than 2 minutes instead of the desired 10.

A similar graph appears in Fig. 7 where four 256 x 256 x 8 sensors are assumed. With only 1.17 Gbytes of mass memory, a system without compression could collect only Sensor Events which were 2.5 minutes in length.

Again, the advantages of employing lossless compression before mass memory storage are quite obvious, particularly when large format sensors are anticipated.

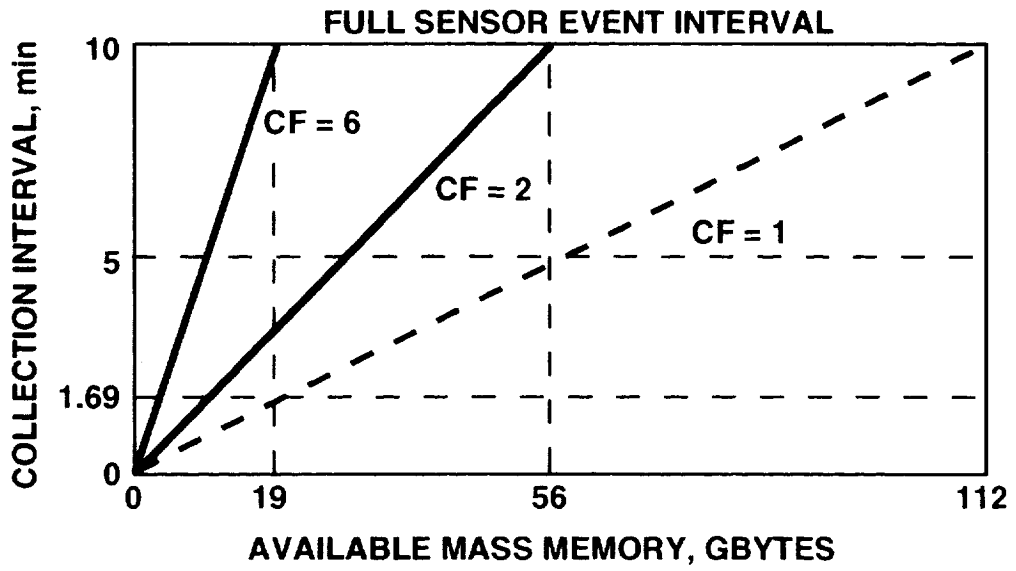


Fig. 6. Mass Memory Impact on Collection Interval, 4 1024 x 1024 x 12 Sensors

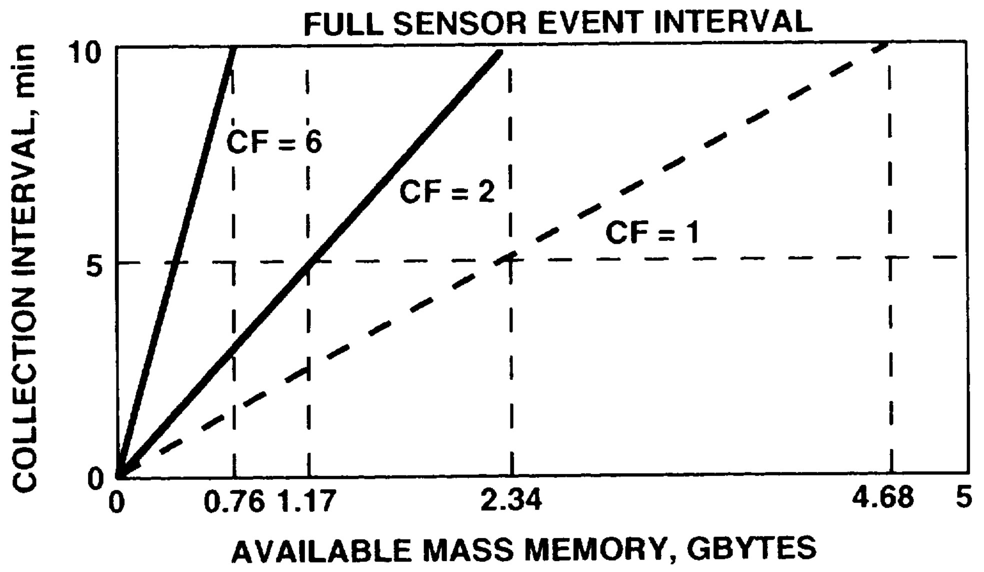


Fig. 7. Mass Memory Impact on Collection Interval, 4 256 x 256 x 8 Sensors

III. DOWNLINK CONSIDERATIONS

ASSUMPTIONS

Ray Rew of Spectrum Astro has specified the MSTI 3 downlink as an uncoded link with the following description:

- Downlink Rate = 1 Mbit/s
- Minimum Error Rate = 10^{-6}

Based on this description of the downlink, a technique for upgrading this capability to up to 3 Mbits/s with virtually error-free communication has been proposed. Implementation should be straightforward and could provide another important technology demonstration for BMDO. This approach for incorporating Reed-Solomon encoding is described in Appendix B.

Assuming there is still a possibility for these improvements to be implemented, we will assume

$$1 \leq R_D \leq 3 \text{ Mbit/s} \qquad (5)$$

where R_D is the downlink data rate, and R_D = 1 Mbit/s is the baseline. Error considerations are also described in Appendix B.

Duty Factor

The situation is worse than R_D in Eq. (5) would imply because the link is only available for 10 minutes out of a 90 minute orbit. The net result is an "effective" average downlink rate of only

$$R_D' = R_D/9 \; (111 \leq R_D' \leq 333 \text{ Kbits/s}) \qquad (6)$$

Table 2 illustrates the extent of the disparity between the (effective) downlink rate, R'_D, and the collection rate, which lists values for the fraction

$$\frac{\text{Collection Rate}}{R_D'} \qquad (7)$$

Table 2. Collection Rate vs. Effective Downlink Rate

		COLLECTION RATE / R_D' (EFFECTIVE DOWNLINK RATE)	
SENSOR TYPE	R_D, Mbits/s	1 SENSOR	4 SENSORS
256 X 256 X 8	1	141	566
	3	47	189
1024 X 1024 X 12	1	3397	13,590
	3	1132	4530

LOSSLESS TRANSMISSION TIMES

Transmission Time

We can begin looking at some of the trade-offs that will need to be considered. Figure 8 illustrates the "required" transmission times (i.e., using R_D in Eq. (5), not R_D' in Eq. (6)) to transmit **selected subsets** of all the frames of a 10 minute Sensor Event, assuming four 1024 x 1024 x 12 sensors. This corresponds to the, possibly conservative, worst case situation in Table 1 (112 Gbytes).

The ordinate in Fig. 8 shows the number of hours of **link time** to return all the data of a central, square 2^n x 2^n sub-image. The abscissa shows the size of one side of this central sub-image, 2^n. This axis can also be viewed as the size of a smaller format image to be sent (not a sub-image).

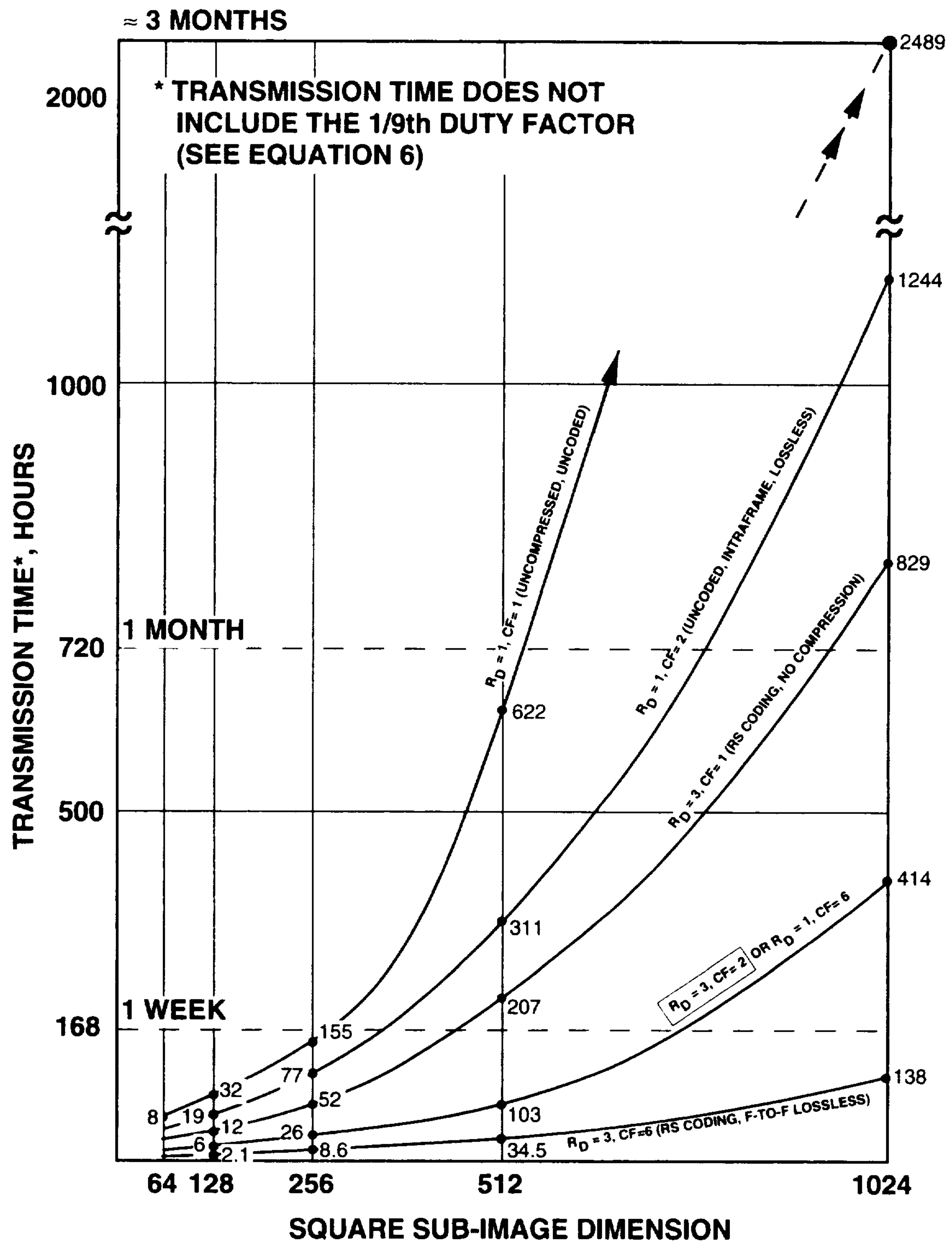


Fig. 8. Transmission Time (active link time) required for transmission of full 10-minute Sensor Events using 4 sensors and 1024 x 1024 x 12 images or central, $2^n \times 2^n$ sub-images.

Cases Considered

The following six assumptions are made about the data compression/downlink capabilities:

a) Data is transmitted uncompressed on a downlink with 1 Mbit/s capability. The graph is indicated as ($R_D = 1$, CF = 1). (8)

b) The MSTI 3 experimental data compression of 2:1 is assumed and the downlink remains at 1 Mbit/s ($R_D = 1$, CF = 2). (9)

c) Data is transmitted uncompressed, but the proposed improved downlink capability using Reed-Solomon coding achieves a downlink data rate of 3 Mbits/s ($R_D = 3$, CF = 1). (10)

d) Assumptions (9) and (10) are combined (R_D =3, CF = 2) for a system gain of 6:1. (11)

e) A 1 Mbit/s downlink is assumed but a 6:1 F-to-F or lossless compression is achieved ($R_D = 1$, CF = 6). This option has the same system gain as Assumption (11), but is probably more difficult, and does not apply to all sensors. (12)

f) Assumptions (10) and (12) are combined for an overall system gain of 18:1 ($R_D = 3$, CF = 6). (13)

Worst Case. The worst case situation occurs when $2^n = 1024$, corresponding to sending all of the 1024 x 1024 images. With the basic ($R_D = 1$, CF = 1) system, 2489 hours of downlink time are required; that is, around 3 months. For the most do-able high performance capability in Assumption (11), this is reduced to around 414 hours (≈2 weeks). In the ideal case in Assumption (13), this might be reduced further to about 6 days.

Reducing 2^n. Reduction in image size has a dramatic effect since total data accumulation is proportional to $(2^n)^2$. Reduction down to the current baseline image size of 256 x 256, reduces the total time to 155 hours. ($R_D = 1$, $CF = 1$, $n = 256$.)

But applying the quite feasible improvements in Assumption (11) reduces this time further to 26 hours, or about 1 day ($R_D = 3$, $CF = 2$, $n = 256$).

Using these same improvements and sending only a central 64 x 64 image, would require only 1.6 hours.

Total Transmission Time (with duty factor)

Unfortunately, link availability is not continuous as indicated in Eqs. (6) and (7). All the times indicated in Fig. 8 must be multiplied by 9 to get the actual duration required to get the same data down. A corresponding graph is shown in Fig. 9.

Clearly, the problems are severe and shouldn't be taken lightly, particularly if large format sensors are really anticipated. With the basic implementation ($R_D = 1$, $CF = 1$) and an image size of 1024 x 1024, the total time necessary is over 2.5 years, making transmission quite impractical. Even assuming 256 x 256 size sensors would only reduce this down to about 2 months.

The most feasible implementation in a near-term time frame is the ($R_D = 3$, $CF = 2$) combination. This gets the maximum time down to about 5 months.

If the "average" 2^n x 2^n frame size were $2^n = 512$, the ($R_D = 3$, $CF = 2$) system could get all the data back in a little over 1 month, perhaps an acceptable situation. This goes to about 1-1/2 weeks if the frame size is instead, the baseline 256 x 256. Using only 8 bits/pixel, reduces this further to about 1 week.

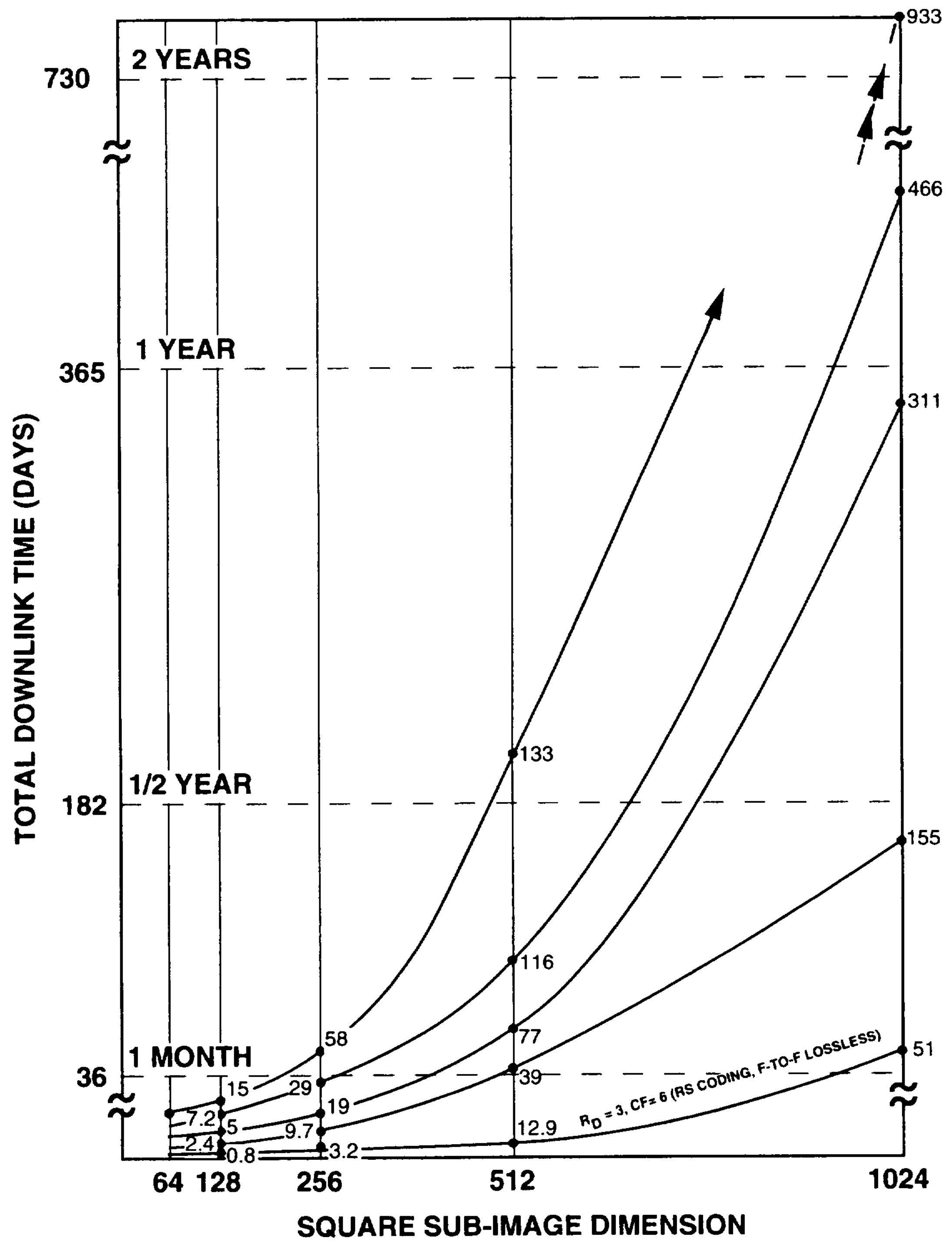


Fig. 9. Total Downlink Time (including 1/9th duty factor) required for transmission of full 10-minute Sensor Events using 4 sensors and 1024 x 1024 x 12 images or central, 2^n x 2^n sub-images

BACKGROUND/FOREGROUND MODES

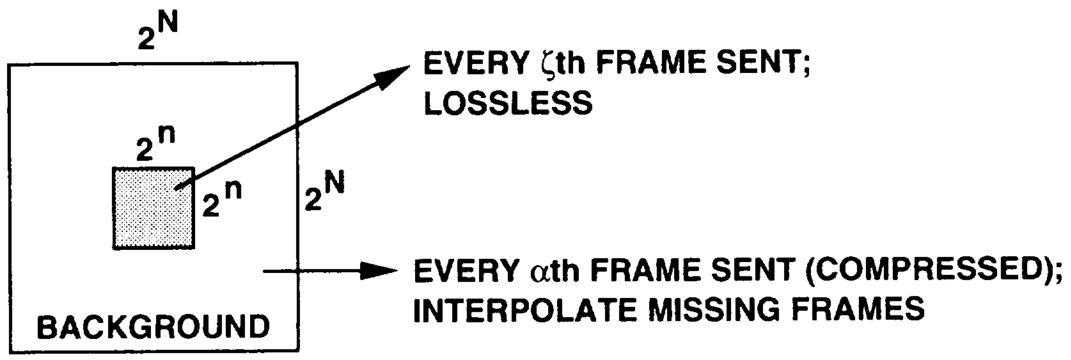
Thus far, all compression considerations have employed only lossless coding, focusing primarily on getting all the data back, unmodified, thus satisfying all the desired goals noted in Paragraphs (1a) through (1e) and (2a) through (2e). This section focuses on situations where either all the data is not required and/or the reconstructed data is not required to have perfect fidelity. Under the following conditions, significantly higher compression factors should be possible.

- Verification;
- Quick survey (browsing) to discover where a problem developed (for subsequent transmission of such data with lossless fidelity);
- Verification and full-frame visual playback.

Definitions

The concept of Background/Foreground Modes (BGFGs) is illustrated in Figs. 10 and 11. To parameterize the discussion:

- A full frame size is $2^N \times 2^N$ (baseline $N = 8$, maximum $N = 10$). (14)
- The central (foreground) $2^n \times 2^n$ region of every ζth frame in a Sensor Event **is transmitted** and reconstructed without error (lossless). (15)
- Full frames are communicated only every αth frame, where $\alpha \geq \zeta$. These are **Background Frames.** Reconstruction fidelity of Background Frames is **not necessarily** lossless. Data not communicated is reconstructed by interpolation from transmitted data. Note that KHILS is equipped for a variation of this in which intervening missing frames are reconstructed by repeating the most recent fully reconstructed frame. (16)



2^N
2^n
2^n
2^N
BACKGROUND
EVERY ζth FRAME SENT; LOSSLESS
EVERY αth FRAME SENT (COMPRESSED); INTERPOLATE MISSING FRAMES

Fig. 10. BGFG Mode Concept.

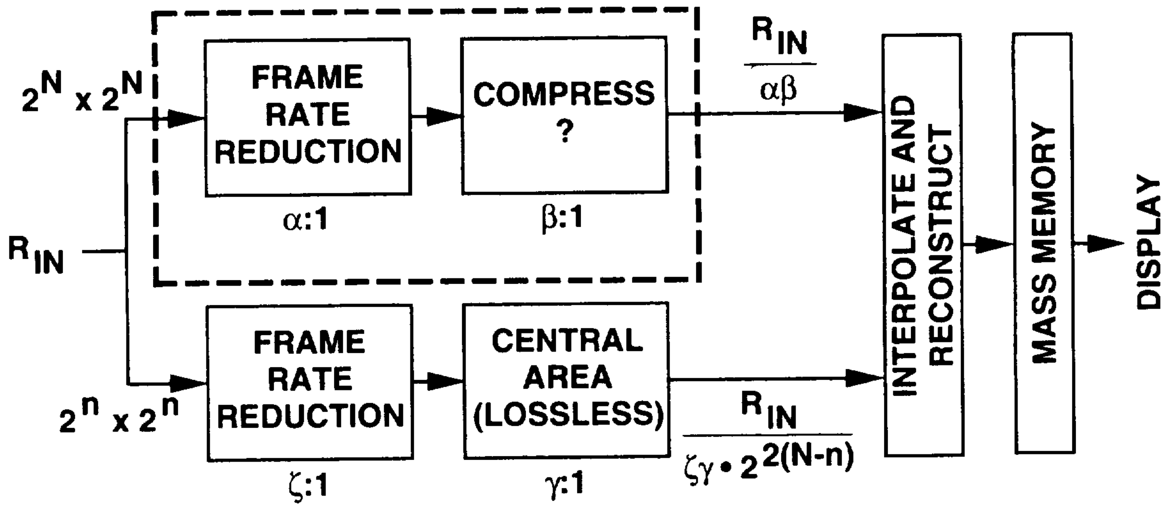


Fig. 11. BGFG Block Diagram

Figure 11 can be used to develop an overall expression for a **System Compression Factor.** Here, R_{IN} denotes the data rate associated with uncompressed data (all the frames).

Foreground compression of the central $2^n \times 2^n$ sub-image area is a two step process:

- A ζ:1 reduction because only every ζth frame is sent, and
- A γ:1 reduction by compressing these central frames.

We will carry these definitions into the specification of an overall System Compression Factor, but then restrict ζ to $\zeta = 1$ in evaluating the consequences of varying parameters. That is, we will later assume that all central frames are sent (full frame rate).

Note also that we restrict the compression of central frames to be lossless. More generally, the compression of the central region could be achieved by any approach, such as those considered for background frames. The System Compression Factor equation below supports that generality, although it would not appear to have any significant value for the MSTI driven goals.

Foreground $2^n \times 2^n$ lossless compression by a factor of γ:1 results in a rate of

$$\frac{R_{IN}}{2^{2(N-n)} \cdot \gamma \cdot \zeta} \tag{17}$$

If we choose $n = 6$, $N = 8$ as realistic baseline parameters, this term becomes

$$\frac{R_{IN}}{16 \cdot \gamma \cdot \zeta} \tag{18}$$

where the central area transmitted is 64 x 64.

Background Frame reduction in rate is a two step process:

- An α:1 reduction because only every αth frame is sent, and
- A β:1 reduction by compressing each of these frames.

The net result is a rate of

$$\frac{R_{IN}}{\alpha\beta} \tag{19}$$

which can be reduced slightly by not "redundantly" sending the central area in each of these frames (which is already sent with perfect quality).

Using these terms we can calculate an overall System Compression Factor as

$$sCF = \frac{1}{\frac{1}{\gamma\zeta}\left(1 - \frac{\gamma\zeta}{\alpha\beta}\right)\left(\frac{1}{2^{2(N-n)}}\right) + \frac{1}{\alpha\beta}} \tag{20a}$$

where the $(1 - \gamma\zeta/\alpha\beta)$ term takes into account this central redundant area, assuming that α divides ζ.

Henceforth, we will assume that $\zeta = 1$ (no frame rate reduction on the central area). Then Eq. 20a reduces to

$$sCF = \frac{1}{\frac{1}{\gamma}\left(1 - \frac{\gamma}{\alpha\beta}\right)\left(\frac{1}{2^{2(N-n)}}\right) + \frac{1}{\alpha\beta}} \tag{20b}$$

A Limiting Case. Note that if we take $\alpha = 1$ (no frame rate reduction on background) but make the compression factor β infinite, sCF in Eq. (20b) reduces to

$$sCF = \gamma \cdot 2^{2(N-n)} \tag{21}$$

This is the overall reduction obtained by discarding all data outside the central $2^n \times 2^n$ area and compressing the center by the factor γ.

Simplification. When $\alpha\beta \gg \gamma$, Eq. 20b can be simplified to

$$sCF = \frac{2^{2(N-n)}}{\frac{1}{\gamma} + \frac{2^{2(N-n)}}{\alpha\beta}} \tag{22}$$

Again, if $\alpha\beta \gg 2^{2(N-n)}$, sCF converges to Eq. (21).

Analysis Assumptions

The following section will use graphs and tables to describe the potential benefits of BGFG modes. We make the following assumptions.

<u>Lossless Compression</u>

- $\gamma = 2$ for lossless intraframe compression of any foreground area. (23)
- $\beta = 2$ for lossless intraframe compression of any background frame. (24)
- $\gamma = 6$ for lossless F-to-F compression of data in which adjacent frames (in time) are separated only by 1/30th second. Thus, we are assuming that F-to-F compression is not applicable to the compression of Background Frames unless $\alpha \rightarrow 1$. (Actually, F-to-F compression does not instantly become useless but the achievable compression factor will decrease as the frames become more separated in time. We make this $\gamma = 6$ restriction here because of uncertainty and because we are focusing on results where $\alpha \gg 1$.) (25)

It should be understood that the possibility of achieving the $\gamma = 6$ F-to-F compression is highly sensor dependent. The actual values achieved on applicable sensors could be more or less than the $\gamma = 6$ assumption used in these analyses.

A word about unnecessary overhead. JPL recently implemented a separate demonstration of lossless compression for the MSTI 3 environment. In this demonstration, the format specification requires the use of redundant 48-bit identifiers on each line, as well as additional lines of engineering data at the end of each frame. Clearly, these types of constraints increase the overhead and decrease the net compression efficiency that can be obtained. The penalty in loss of efficiency is quite dramatic when frame sizes are small, as BGFG modes where the central area is transmitted could be as small as 64 x 64. The following analysis illustrates the magnitude of these penalties.

Let the image size be $2^n \times 2^n$ with a quantization of q bits/sample and let γ be the compression factor achieved on the actual image as we have been assuming. Then the number of bits/pixel required on the actual data is given by

$$\frac{q}{\gamma} \tag{26}$$

but the overhead adds a penalty of

$$\frac{12 + q}{2^{n-2}} \text{ bits/pixel} \tag{27}$$

for a total of

$$\frac{q}{\gamma} + \frac{12 + q}{2^{n-2}} \text{ bits/pixel} \tag{28}$$

Equation 27 is plotted in Fig. 12 for q = 12 and q = 8.

- With a quantization of q = 12, the penalty when compressing a 64 x 64 size image (n = 6) is as much as 1.5 bits/pixel. Instead of representing an image with 6 bits/pixel (with $\gamma = 2$), the representation would require 7.5 bits/pixel, a **25%** increase. (29)

- If quantization were q = 8, a 2:1 compression would require 4 bits/pixel for the real data, but 5.25 bits/pixel for image plus overhead. In this case the overhead incurs a penalty of **31%**. (30)

- Now suppose that q = 12 but F-to-F representation yielded compression of $\gamma = 6$ (other constraints would need to be changed too). Then, $\gamma = 6$ means that only 2 bits/pixel are required for the image data but 3.5 bits/pixel are required for image plus overhead. The overhead now incurs a **75% penalty**. (31)

Paying such a penalty is absurd. **Subsequent discussions will assume that these issues have been resolved so that an overhead penalty is negligible for all n**.

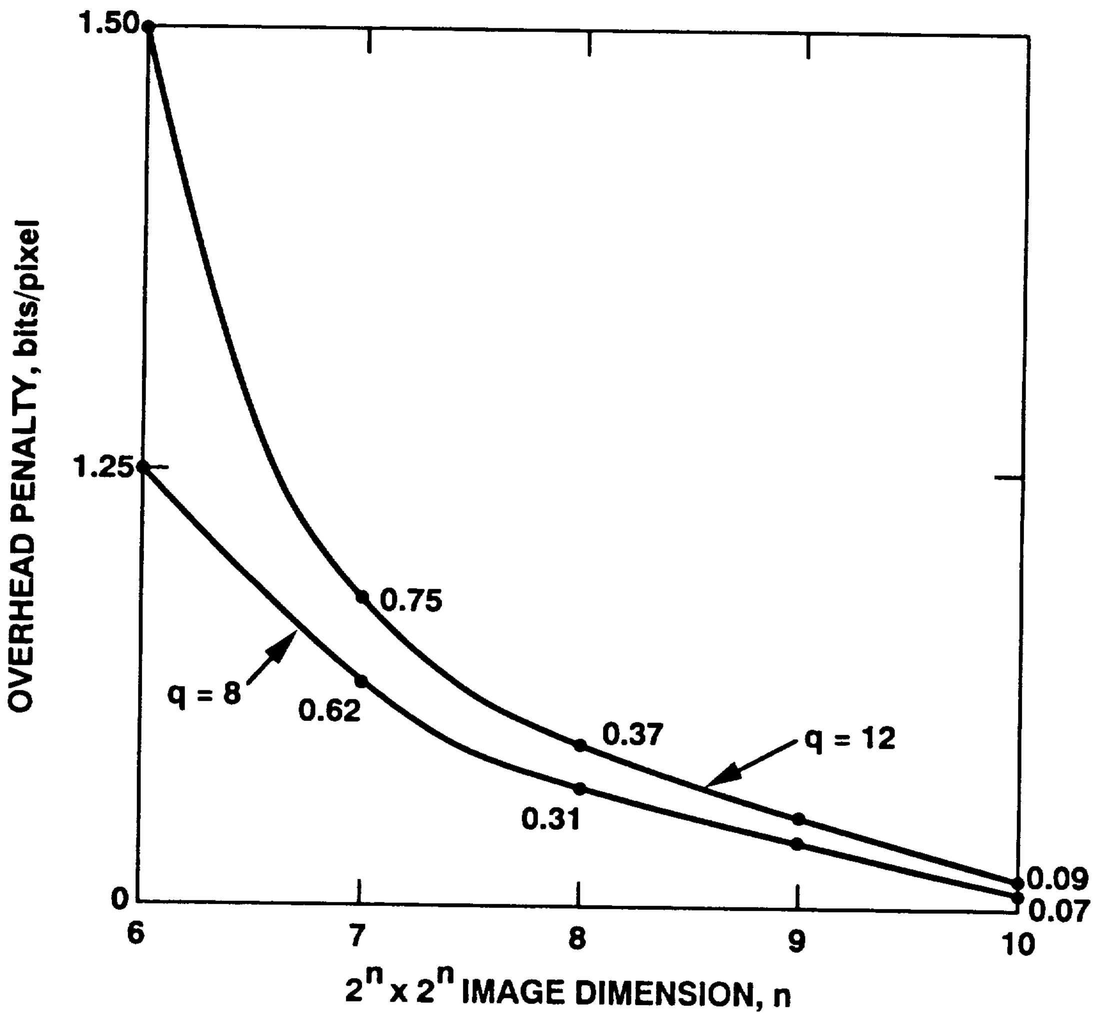


Fig. 12. Overhead Penalty

Background Frames

Setting parameters for Background Frames is more difficult since we are now allowing "lossy compression." Quality of reconstruction is based on a visual observation of reconstructed frames (including reconstructed missing frames), usually displayed at a full frame rate (30 f/s). Specifying quality is particularly difficult because, by definition, the primary central region is always perfect. Further, since the net reduction factor is $\alpha\beta$, the same reduction factor can be achieved by various combinations of α and β with varying results.

But we can make certain practical assumptions on techniques and technology that will allow us to "see" the approximate consequences of pursuing further developments that might offer benefits in later flights.

- **Lossless** – $\beta = 2$ (see paragraphs 24 and 25). (32)

- **JPEG** – We will assume that operation of the JPEG algorithms can be acceptably achieved with a compression factor of $\beta = 10$. (33)

- **FRACTALS** – We will assume $\beta = 100$ when using FRACTAL-based algorithms. (34)

Certainly, these assumptions are subject to revision, but they provide a likely range of acceptable lossy compression factors. The FRACTAL and JPEG approaches are identified as being likely candidate compression algorithms that might achieve the two ends of this range. The $\beta = 10$ factor might be too low for JPEG and the $\beta = 100$ factor might be too high for FRACTALS.

Note that in this study we are excluding the possibility of extremely simple lossy approaches which achieve a high β by combining sub-sampling with lossless compression. While clearly inferior to JPEG and FRACTALS, such simplicity might still accomplish objectives more easily in a hardware-constrained implementation.

BGFG Analysis

It is difficult to consider all the possibilities one might consider. In the following discussion we will assume that with foreground frames of size $2^n \times 2^n$ and background frames of size $2^N \times 2^N$ (see Fig. 10)

$$N = n + 2 \qquad (35)$$

so that

$$2^{2(N-n)} = 16 \qquad (36)$$

in Eqs. 20 and 22. Then sCF in Eq. (22) reduces to

$$\mathrm{sCF} = \frac{16}{\frac{1}{\gamma} + \frac{16}{\alpha\beta}} \tag{37}$$

for the BGFG combinations in Table 3.

Table 3. BGFG Image Sizes Studied

BACKGROUND IMAGE SIZE	FOREGROUND IMAGE SIZE
1024 X 1024	256 X 256
512 X 512	128 X 128
256 X 256	64 X 64

Figure 13 plots the System Compression Factor sCF from Eqs. (20) and (37) as a function of the Background Frame Rate Reduction Factor, α, for various assumptions on algorithm usage and using Table 3.

γ = 2, β = 2. Here baseline intraframe lossless coding is applied to both Foreground and Background frames. This is really the baseline concept. With $\alpha = 1$ (no frame rate reduction), sCF = 2. As α increases, the impact on sCF decreases, because the fraction of total rate generated by background frames decreases. sCF = 16 at $\alpha = 16$ ($\approx$2 f/s) but doubling α to 32 only increases sCF by about 30%.

γ = 6, β = 2. In this case we apply F-to-F lossless compression to the foreground frames with the assumption that $\gamma = 6$. With a strict adherence to Eq. (14), sCF $\rightarrow \beta = 2$ as $\alpha \rightarrow 1$. But this is not realistic since as $\alpha \rightarrow 1$ the frame rate approaches 30 f/s and background frames really become foreground frames, allowing the use of F-to-F techniques on all data. This is indicated by the solid arrow.

At $\alpha = 8$ ($\approx$4 f/s) the superior 6:1 compression on foreground frames improves sCF by only a small amount. This is because at $\alpha = 8$, the contribution from background frames (at $\beta = 2$) dominates.

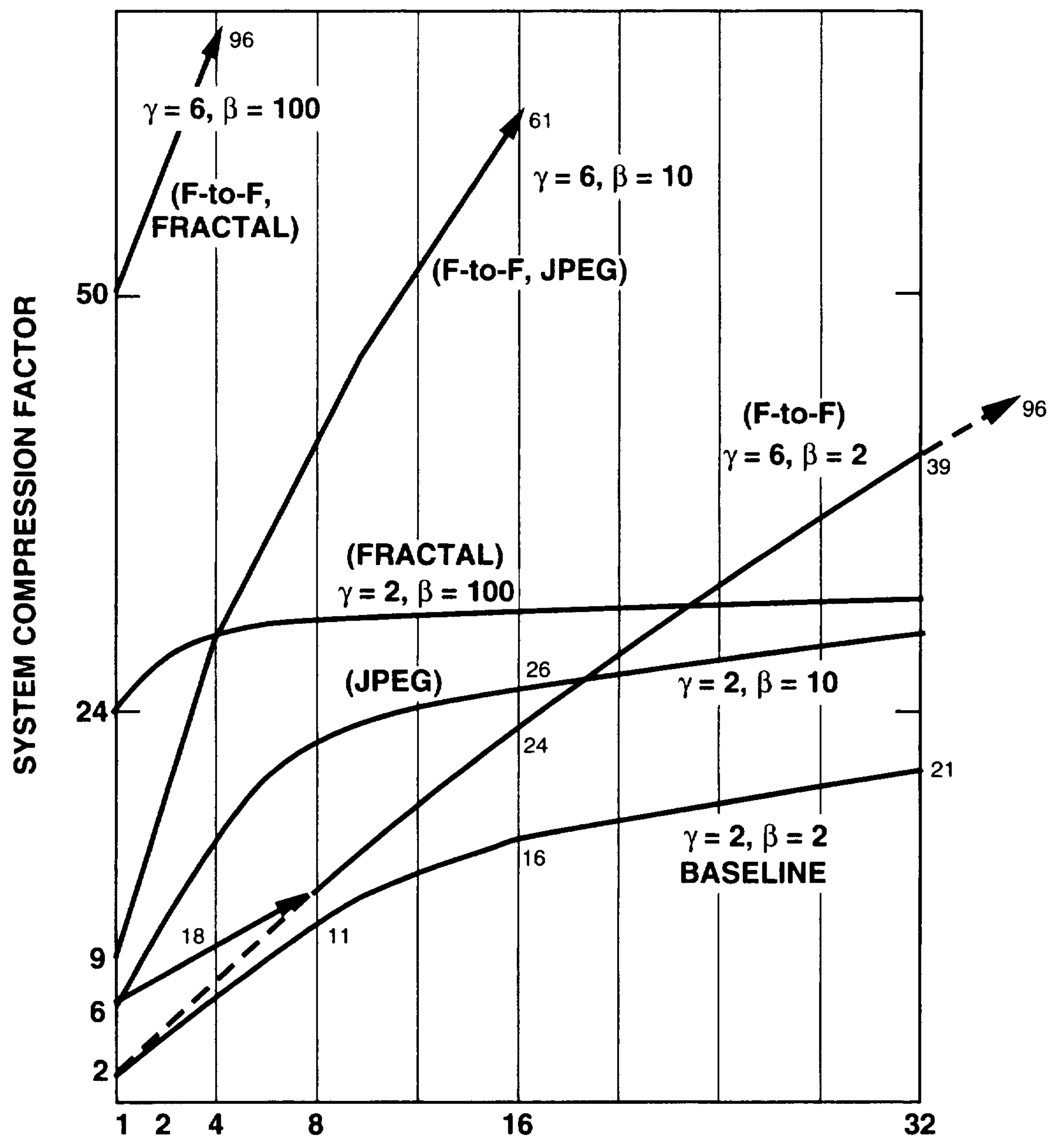


Fig. 13. BGFG System Compression Factors using Table 3 Image Sizes

Even at $\alpha = 16$ (≈2 f/s) the overall sCF is increased by only 50% over the use of intraframe compression alone.

$\gamma = 2$, $\beta = 10$ (JPEG). Now apply JPEG at $\beta = 10{:}1$ on the background frames and maintain a $\gamma = 2{:}1$ ratio on the foreground frames. Surprisingly, at $\alpha = 16$ the net sCF is almost identical to the one obtained by using the $\gamma = 6$ F-to-F compression (with $\beta = 2$). When $\alpha \rightarrow 32$, this mode is significantly worse than with $\gamma = 6$, $\beta = 2$.

$\gamma = 2$, $\beta = 100$ (FRACTAL). Again the results are surprising. Except at low frame rate reductions ($4 \leq \alpha \leq 8$), the advantage of even 100:1 on background frames doesn't materialize in large improvements in sCF. At $\alpha = 16$, this mode is only slightly better than $\gamma = 6$, $\beta = 2$ and perhaps twice as good as the baseline ($\gamma = 2$, $\beta = 2$).

$\gamma = 6$, $\beta = 10$ (JPEG). By introducing F-to-F compression on the foreground frames, a dramatic improvement in sCF is obtained at the lower values of α. Or interpreted another way, F-to-F improvements to the foreground lossless compression factor net an equivalent improvement in sCF, when using JPEG on background frames.

$\gamma = 6$, $\beta = 100$ (FRACTAL). As with JPEG, real improvements occur. Nearly 100:1 is obtained at a very low $\alpha = 4$.

Key Points on sCF. Some key observation points are provided in Tables 4 and 5.

If we take into account the results in Fig. 13 and Tables 4 and 5, and the downlink assumptions in Equations (4)–(6), we can compute the potential value of these modes in speeding up downlink transmission times. Tables 6 and 7 provide selected results which specify the Actual Sensor Event downlink times (i.e., including the 1/9th duty factor) assuming image sizes of $2^N \times 2^N$ where $N = 8$ or $N = 10$ and $R_D =$ 1 Mbit/s or $R_D = 3$ Mbits/s.

The results for 1:1, 2:1 and 6:1 (Table 6), taken directly from Fig. 9, correspond to full transmission of all Sensor Data in a lossless fashion ($\alpha = 1$).

The remaining results, in Table 7, assume a frame rate reduction factor of $\alpha = 8$ ($\approx$4 f/s).

Table 4. Important BGFG Mode Observations, I.

α	PARAMETERS	WHEN	SYSTEM CF
∞	$\gamma = 2$, N.A.	VERIFICATION	32
1	$\gamma = 2$, $\beta = 2$ (BASELINE)	DATA SET GENERATION	2
ALL DATA SENT	$\gamma = 6$, $\beta = 6$ (F-to-F)	DATA SET GENERATION	6
	$\gamma = 2$, $\beta = 10$ (JPEG)	VISUAL	6
	$\gamma = 2$, $\beta = 100$ (FRACTAL)	VISUAL	24
	$\gamma = 6$, $\beta = 100$ (F-to-F, FRACTAL)	VISUAL	50

Table 5. Important BGFG Mode Observations, II.

α	PARAMETERS	WHEN	SYSTEM CF
8	$\gamma = 2$, $\beta = 2$ (BASELINE)	BASELINE, VISUAL	11
	$\gamma = 2$, $\beta = 10$ (JPEG)	LOSSY VISUAL	22
	$\gamma = 2$, $\beta = 100$ (FRACTAL)	LOSSY VISUAL	31
	$\gamma = 6$, $\beta = 2$ (F-to-F)	LOSSLESS VISUAL	14
	$\gamma = 6$, $\beta = 10$ (F-to-F, JPEG)	LOSSY VISUAL	44
	$\gamma = 6$, $\beta = 100$ (F-to-F, FRACTAL)	LOSSY VISUAL	93

Table 6. BGFG Actual Downlink Times Default α = 1 (all data lossless)

IMAGE DIMENSION 2^N	BGFG ACTUAL DOWNLINK TIME (DAYS)					
	sCF = 1:1		sCF = 2:1		sCF = 6:1	
	R_D=1	R_D=3	R_D=1	R_D=3	R_D=1	R_D=3
256	58	19	29	9.7	9.7	3.2
1024	933	311	466	155	155	51
	NO COMPRESSION, FULL RECONSTRUCTION		BASELINE, FULL RECONSTRUCTION		F-TO-F, FULL RECONSTRUCTION	

Table 7. Selected BGFGs, Actual Downlink Times, α = 8. Image Size $2^N \times 2^N$, Central Area Size $2^{N-2} \times 2^{N-2}$

IMAGE DIMENSION 2^N	BGFG ACTUAL DOWNLINK TIME (DAYS)									
	sCF = 11		sCF = 16		sCF = 30		sCF = 44		sCF = 93	
	R_D=1	R_D=3	R_D=1	R_D=3	R_D=1	R_D=3	R_D=1	R_D=3	R_D=1	R_D=3
256	21	7	14	4.8	7.8	2.6	5.3	1.8	2.5	0.83
1024	84	28	58	19.4	31	10.4	21.2	7.1	10	3.3
	$\gamma = 2, \beta = 2$; ALL LOSSLESS		$\gamma = 2, \beta = 10$ (JPEG)		$\gamma = 2, \beta = 100$ (FRACTAL)		$\gamma = 6, \beta = 10$ (JPEG)		$\gamma = 6, \beta = 100$ (FRACTAL)	

Major Observations

We can now make some major observations.

If Lossy Compression Is Not Employed:

a) Improved F-to-F lossless compression would be valuable for sending all data back ($\alpha = 1$).

b) There is little advantage in using F-to-F lossless compression rather than BGFG Modes unless $\alpha \geq 16$.

If Lossy Compression Is Used on Background Frames:

The net gains are significantly constrained by a 2:1 foreground compression factor for $\alpha \geq 8$. Improvements by using F-to-F map directly into improvements in overall sCF.

The most significant system benefit from future improvements to compression for MSTI (i.e., beyond the baseline 2:1 lossless intraframe compression) would come from F-to-F lossless. It is important to determine the feasibility of achieving the "estimated" 6:1 F-to-F CF before embarking on costly developments involving lossy compression factors. Improvements via F-to-F lossless would impact all aspects of the data system: mass memory, downlink for all the data and downlink in conjunction with BGFG modes.

INDEXING THE DATA

Suppose the most sophisticated BGFG mode just discussed ($\gamma = 6$, $\beta = 100$, $\alpha = 8$) were used to rapidly communicate a Sensor Event down (sCF = 93:1). If all goes well, the target remains in the center where the data is perfect and the quality of surrounding (interpolated) data is high enough to provide a quite adequate full frame display. If something goes wrong, the display on the ground should be good enough to allow detection of the problem.

But to determine why a problem developed, all the image data (full frames, perfect fidelity, full frame rate) should be available for a time slice around the point where a problem was detected. This is the **Critical Data** (see paragraph 2b).

It makes sense to secure this Critical Data before transmitting all the data to avoid the consequence of a potential system/spacecraft failure impacting the data. To do this means that the mass memory storage of Sensor Events cannot be treated as simply storing one long string of data. Instead, there needs to be a means to both command and access specific segments of the stored data (as determined by observing reconstructed BGFG data on the ground).

As an example, suppose the size of the Critical data segment was 30 seconds (x 1000 frames). Compared to the communication of all data from a full 10 minute Sensor Event, this provides **a 20:1 reduction** in the time required to secure the Critical Data (a worst case situation). If the problem only developed on one of four sensors, the effective reduction is 80:1 (although other sensor data might provide supporting evidence of the problem).

IV. SUMMARY

Much of the information provided here is conceptual and cannot be adequately summarized. Graphs, tables and concepts need to be studied and understood. However, we can make numerous observations that may be helpful to keep in mind.

1. An initial program emphasis on lossless intraframe compression makes sense. This type of compression appears feasible in a short time frame and can significantly improve all aspects of MSTI storage and communication. Therefore, it would be desirable to initially demonstrate as many aspects of its usage as possible:

   - Compression into memory
   - Compression out of memory
   - Compression within BGFG Modes

2. Investigations of F-to-F lossless compression might yield valuable performance improvements for later flights – affecting all aspects of MSTI storage and communication.

3. BGFG Modes – a wide range of investigation is possible. Such modes will be most valuable if F-to-F lossless compression can be effectively used.

4. Ground processing/decompression/display/command considerations need to be integrated into the overall end-to-end problem thinking.

   - Interactively responding to received data from a Sensor Event would require:

     a) a fast decompressor
     b) some form of on-board image indexing

5. Using FRACTALS or JPEG requires uncompressed data from mass memory. If lossless compression were used to improve mass memory, an on-board decompressor would be needed.

- Data compressed into mass memory with lossless coding might be used directly in BGFG Modes without a decoder, but would need additional formatting.

6. Appendix B provides an apparently quite feasible, relatively low-cost improvement to the MSTI data link – this would include a possible 3:1 improvement in data rate and improved error characteristics for error-sensitive data (e.g., compressed). This implementation would additionally provide the Ballistic Missile Defense Organization with another important technology demonstration with possible broad impact on other programs.

APPENDIX A

Algorithms for High-Speed Universal Noiseless Coding

I. INTRODUCTION

References 1–4 provide the development and analysis of some practical adaptive techniques for efficient noiseless (lossless) coding of a broad class of data sources. These have been applied, in various forms, to numerous applications.

Those functions and algorithms most desirable for incorporation in a "coding module" which could be implemented using current custom VLSI capabilities were presented at the first NASA Data Compression Workshop at Snowbird, Utah, in 1988.[5] A workshop committee recommended that NASA should proceed and implement this "coding module." Since then, both the Jet Propulsion Laboratory (JPL) and the Microelectronics Research Center (MRC) at the University of New Mexico have implemented first-generation single-chip CMOS VLSI coding modules.[6-9] The MRC 1.0 μm coding chip was successfully tested under laboratory conditions at up to 900 Mbits/s.[7-9] A companion MRC "decoding module" is designed to run at up to half the maximum rate of the coding module. These first-generation MRC chips are now available commercially from Advanced Hardware Architectures in Moscow, Idaho.[10]

Both the MRC and JPL developments are nearing completion of second-generation space-qualified versions for the coding modules.[11-14] It is anticipated that the high performance functionality of these modules, first- and second-generation, can serve most of NASA's science data needs where a lossless representation is appropriate.

The intent of this paper is to provide a concise description of the basic algorithmic and performance characteristics which are embodied in the coding modules. A more general algorithmic development can be found in Ref. 15 along with some application notes. Observe that the actual implementations have diverged slightly from the definitions provided here and from each other. Most of these subtleties will be discussed.

II. THE CODING MODULE

A functional block diagram of a general-purpose lossless "coding module" is shown in Fig. 1. A variation in Rice's original notation (of subscripting the Greek letter ψ) will be used to name various coding operations. Subsequent sections will quickly converge to more specific definitions that relate to the VLSI modules being implemented.

The input to this coding module

$$\tilde{X}^n = x_1\, x_2 \ldots x_J \tag{1}$$

is a J sample block of n bit samples. $\tilde{Y}$ is a priori or Side Information that might help in the coding process.

The **overall** unspecific process of representing $\tilde{X}^n$ is named PSI?+ so that the actual coded result is

$$\text{PSI?} + [\tilde{X}^n, \tilde{Y}]$$

As Fig. 1 shows, the coding process is split into two independent steps discussed below.

Step 1

A Reversible Pre-processor is a process designed to convert the source represented by $\tilde{X}^n$ sequences (and $\tilde{Y}$) into a close approximation to a STANDARD FORM Data Source, represented by $\tilde{\delta}^n$ sequences. This process usually includes a de-correlation procedure (prediction).

The pre-processor converts each $\tilde{X}^n$ (and corresponding $\tilde{Y}$, if any) into

$$\tilde{\delta}^{n'} = \delta_1\, \delta_2 \ldots \delta_J \tag{2}$$

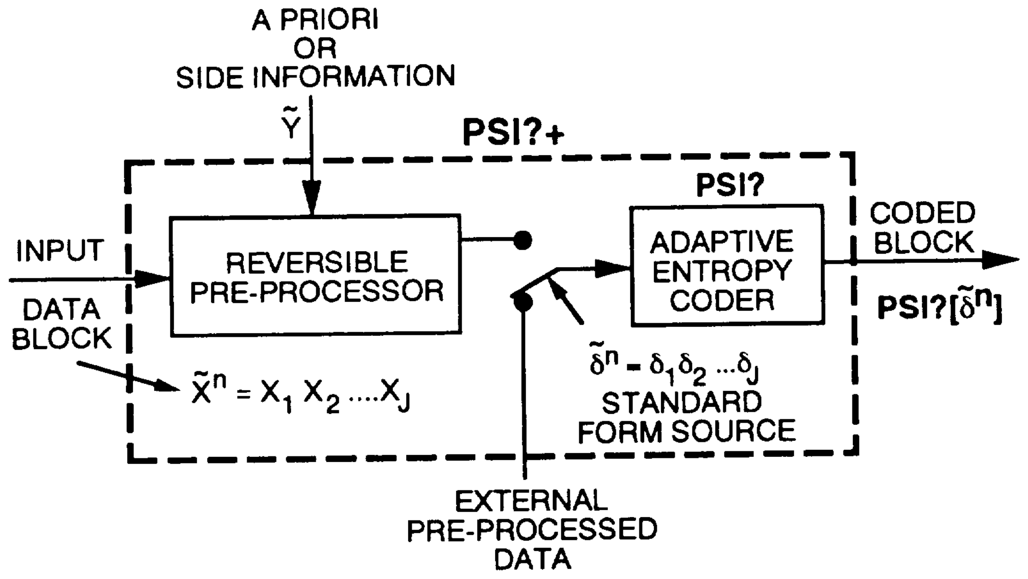


Figure 1. General Purpose Noiseless Coding Module Block Diagram

a $J \geq 1$ sample sequence of n' bit samples. Usually n = n', and we will henceforth assume that here.

Standard Form Source. Specifically,

a) Samples of $\tilde{\delta}^n$ are the non-negative integers 0, 1, 2, ..., q (3)

b) Samples of $\tilde{\delta}^n$ are independent (4)

c) With $p_i = \Pr[\delta_j = i]$, the probabilities are ordered so that the smaller integers occur more frequently, i.e.,

$$p_0 \geq p_1 \geq p_2 > \dots \quad (5)$$

d) The source entropy is given as

$$\overline{H}_\delta = -\sum_i p_i \log_2 p_i \quad \text{bits/sample} \quad (6)$$

The best pre-processor will meet these conditions and produce the lowest $\overline{H}_\delta$.

Step 2

An adaptive entropy coder, named PSI? for now, efficiently represents (Standard Source) pre-processed $\tilde{\delta}^n$ sequences with the coded result

$$\text{PSI?}\,[\tilde{\delta}^n] = \text{PSI?} + [\tilde{X}^n, \tilde{Y}]$$

This entropy coder is independent of the pre-processor. The pre-processor's goal is to achieve performance that remains close to $\bar{H}_\delta$ as it varies with time.

Note that the coding module in Fig. 1 allows for coder PSI? to be used directly on externally supplied pre-processed data.

Prelude to the Details

A general form of an Adaptive Entropy coder (designed to efficiently represent Standard Data Sources), which chooses from multiple algorithm options on a block-by-block basis, will be identified in the next section. Specific sets of such code options will be defined and incorporated in this structure as a **parametrically** defined adaptive coder. In doing so, the unspecific "PSI?" will be turned into a specific coder called "PSIss."

Finally, the specific parameters of PSIss that are used in current VLSI implementations will be identified.

III. ADAPTIVE ENTROPY CODER FOR THE STANDARD SOURCE

PSI? of Fig. 1 represents the general-purpose adaptive coder called PSI11 in Refs. 2-4 and 15. Basically, such a coder chooses one of a set of Code Options (coding algorithms) to use to represent an incoming block of "**pre-processed**" data samples. A unique binary identifier precedes a coded block to tell a decoder which decoding algorithm to use. The following discussion will identify specific code options.

Code Options

Backup. When no coding of any form is performed on the data, we call this PSIbu

$$\text{PSIbu}\,[\tilde{\delta}^n] = \tilde{\delta}^n \tag{7}$$

This representation is used in an adaptive coder when all other available code options fail to compress $\tilde{\delta}^n$. References 7-9 call this the "default" option.

The Fundamental Sequence Code. Recall that the pre-processed samples of $\tilde{\delta}^n$ are the non-negative integers $i \geq 0$. A variable length "Fundamental Sequence Code, fs", is defined for each i as follows

$$fs[i] = \overbrace{0\,0\,0\,.\,.\,.\,.\,.\,0\,0\,0}^{i \quad \text{zeroes}}\,1 \quad \text{for } i \geq 0 \tag{8}$$

That is, simply append a 1 to the end of a sequence of i zeroes.

The "Fundamental Sequence" itself is the application of fs[.] to all the samples of $\tilde{\delta}^n$. Following Rice's notation,[1]

$$\text{PSI1}[\tilde{\delta}^n] = fs[\delta_1] * fs[\delta_2] * \ldots fs[\delta_J] \tag{9}$$

[1] An asterisk, *, is used to emphasize the concatenation of sequences.

is the Fundamental Sequence. This defines Code Option, PSI1.

Split-Sample Modes. The code option definitions here are basically to "split" off the k least significant bits of each $\tilde{\delta}^n$ sample and send them separately. The remaining n-k most significant bit samples are then coded using PSI1. Specifically, with

$$\delta^n = \delta_1 \, \delta_2 \ldots \delta_J$$

Let

$$\tilde{M}^{n,k} = m_1 \, m_2 \ldots m_J \tag{10}$$

be the sequence of all the n-k most significant bit samples of δ^n and let

$$\tilde{L}_k = sb_1 * \; sb_2 * \ldots . \; sb_J \tag{11}$$

denote the corresponding sequence of all the k-bit least significant bit samples of $\tilde{\delta}^n$.

That is

$$\delta_i = m_i * \; sb_i \tag{12}$$

The "Split-Sample" Mode Code Option PSI1,k is defined by

$$PSI1{,}k[\tilde{\delta}^n] = PSI1[\tilde{M}^{n,k}] * \tilde{L}_k \tag{13}$$

Note that k = 0 is a special case where

$$PSI1{,}0 = PSI1 \tag{14}$$

and when k = n

$$PSI1{,}n = PSIbu \tag{15}$$

The Individual Code Words. Note that **the individual code word assigned to** δ_i in (12) when code option PSI1,k is applied is given as

$$fs[m_i] * \ sb_i \tag{16}$$

That is, the Fundamental Sequence Code, fs[·] in (9), is applied to the most significant n-k bits of δ_i, followed by the least significant k bits of δ_i. From this description, it should be easier to see that the **only variable-length-code operation ever required is the application of fs[·]**, since the " sbs" can simply be shifted out, and fs[·] can be implemented **without any table lookups**.

Performance of the Individual PSI1,k Options. Under certain familiar assumptions about the type of data source (these assumptions will be described in a later section),

the individual "variable length codes" represented by (16) were shown by Yeh[17,18] **to be equivalent to Huffman Codes.** (17)

Thus they are not only extremely simple, they are **optimum** too.

But even more important for their application in an adaptive coder, the entropy where PSI1,k achieves its best performance is at

$$\overline{H}_\delta^k \approx k + 2 \text{ bits/sample} \tag{18}$$

and performance remains close to $\overline{H}_\delta^k$ over a range of about ±0.5 bit/sample. Thus there is at least one PSI1,k option that should provide efficient coding for any

$$\overline{H}_\delta > 1.5 \text{ bits/sample} \tag{19}$$

Such conclusions can also be drawn directly from simulations using a broad range of data sources.

Split-Sample Adaptive Coder, PSIss

We can now use the Split-Sample Modes described above to replace the general coder PSI? in Fig. 1 with a specific class of parametrically defined adaptive coders, named PSIss. PSIss is based on the following parameters:

$$\begin{aligned} &J = \text{block size} \geq 1 \\ &n = \text{Input Bits/Sample} \\ &N = \text{number of Code Options} \\ &\lambda \geq 1 \text{ (Dynamic Range Parameter)} \end{aligned} \tag{20}$$

A functional block diagram is shown in Fig. 2.

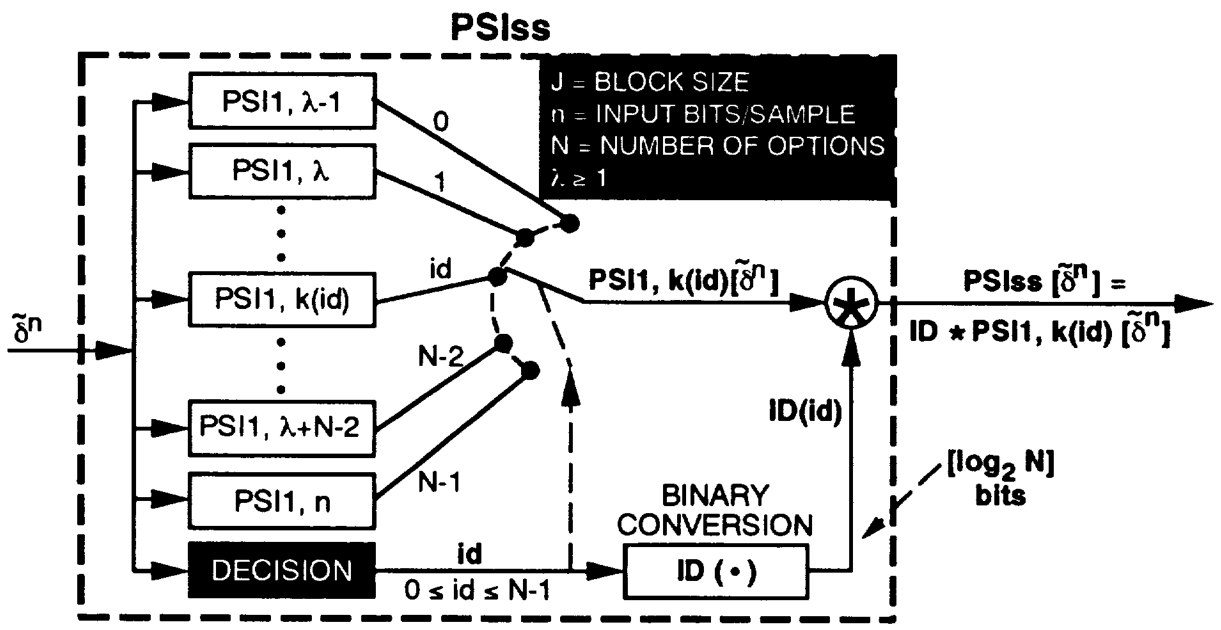


Figure 2. Parametric Split-Sample Adaptive Coder Functional Block Diagram

The representation of J sample $\tilde{\delta}^n$, using an N option PSIss with parameter $\lambda \geq 1$, is given by

$$\text{PSIss}[\tilde{\delta}^n] = \text{ID(id)} * \text{PSI1, k(id)}[\tilde{\delta}^n] \tag{21}$$

where

$$id = 0, 1, 2, \ldots N\text{-}1 \tag{22}$$

is the integer value of a coder identifier for the options used, and ID(id) is its standard binary representation, requiring[2]

$$\lceil \log_2 N \rceil \text{ bits} \tag{23}$$

PSI1,k(id) is the Split-Sample option specified by id, where

$$k(id) = \begin{cases} n \text{ for } id = N\text{-}l \\ \lambda - 1 + id \quad \text{Otherwise} \end{cases} \tag{24}$$

for parameter $\lambda \geq 1$. By (15) and (24) the last option is PSIbu in (7).

Dynamic Range. Except for limiting cases, the range of entropies where PSIss can be expected to efficiently represent pre-processed $\tilde{\delta}^n$ sequences has been shown to be closely specified by

$$\lambda + 0.5 \leq \bar{H}_\delta \leq \min \begin{cases} n \\ \lambda + N - 0.5 \end{cases} \tag{25}$$

A close look at this expression shows that **each increase in λ moves the Dynamic Range of efficient performance upwards by 1 bit/sample.**

Performance Graph. A graph of typical performance for PSIss with N = 12, λ = 1, n = 14 and J = 16 is shown in Fig. 3.

---

2 $\lceil z \rceil$ is the smallest integer, greater than or equal to z.

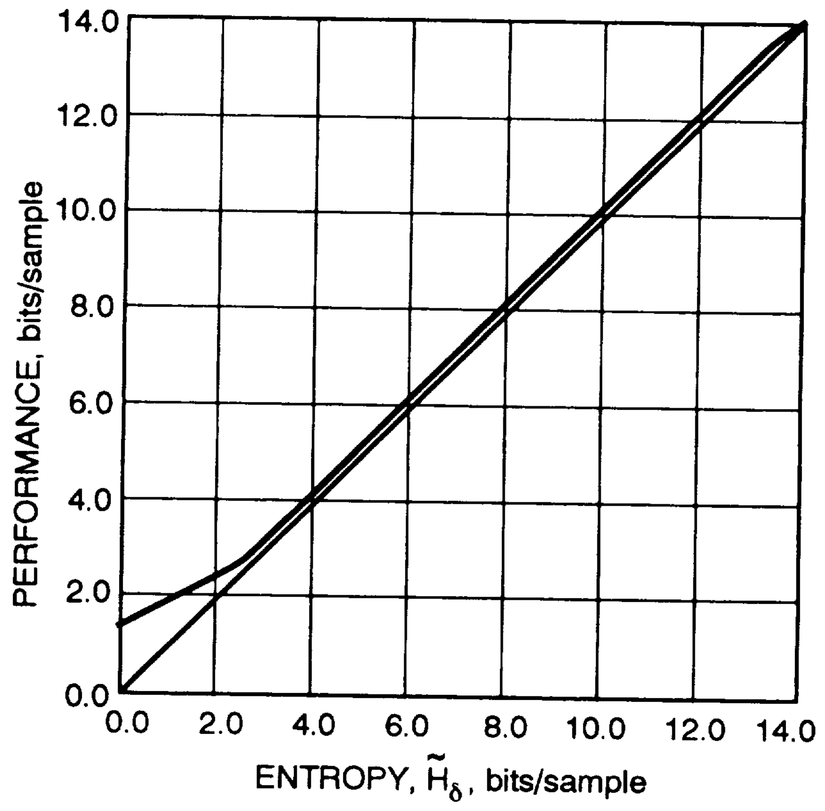


Figure 3. PSIss Average Performance for N = 12, n = 14, λ = 1, J = 16

Choosing the Right Option. The optimum criterion for selecting the best option to use to represent $\tilde{\delta}^n$ is to simply choose the one that produces the shortest coded sequence. That is,

choose k = k* if [3]

$$(\text{PSI1},k^* [\tilde{\delta}^n]) = \min_k \{\mathscr{L}(\text{PSI1},k [\tilde{\delta}^n])\} \qquad (26)$$

Now, letting

$$F_k = \mathscr{L}(\text{PSI1} [\tilde{M}^{n,k}])$$

we have from (12)

[3] $\mathscr{L}(\tilde{Z})$ = length of sequence $\tilde{Z}$ in bits.

$$\mathscr{L}(\mathrm{PSI1},k\,[\tilde{\delta}^n]) = F_k + Jk \tag{27}$$

and from (8) - (10)

$$F_k = \sum_{i=1}^{J} m_i + J \tag{28}$$

(i.e., the sum of the most significant n-k bit samples plus the block size).

Thus (27) and (28) can simplify the decision making in (26) without actually coding the data. But this can be further simplified.

By taking advantage of the randomness in the least significant bits, the expected value of F_k can be related to F_0 by[4,15]

$$E\{\, F_k | F_0 \,\} = 2^{-k} F_0 + \frac{J}{2}(1 - 2^{-k}) \tag{29}$$

which we use as an estimate in (27). We have

$$\begin{aligned} \gamma_{1,k}(\tilde{\delta}^n) &= 2^{-k} F_0 + \frac{J}{2}(1 - 2^{-k}) + Jk \\ &\approx \mathscr{L}(\mathrm{PSI1},k\,[\tilde{\delta}^n]) \end{aligned} \tag{30}$$

We can then choose $k = k^*$ if

$$\gamma_{1,k^*}(\tilde{\delta}^n) = \min_k \{\, \gamma_{1,k}(\tilde{\delta}^n) \,\} \tag{31}$$

and this leads to distinct decision regions based solely on F_0 (which by (28) can be determined by adding up the original samples since $m_i = \delta_i$). The boundaries to adjacent PSI1,k decision regions are given by

$$F_0 = \frac{J}{2} + J\,(2^{k+1}) \text{ bits} \tag{32}$$

Any PSI1,k option will generate more bits than PSIbu when

$$F_0 > \frac{J}{2}\left[\, (n\text{-}k)\, 2^{k+1} + 1 - 2^k \right] \text{ bits} \qquad (33)$$

A sample table of decision regions is shown in Table 1 for an N = 8 option PSIss.

Note that if the largest value of k used is $k = \ell$, then PSI1,ℓ will generate more bits than PSIbu when

$$F_\ell > (n - \ell)\, J \text{ bits} \qquad (34)$$

which may be a simpler test than (33) provides, in some cases.

Table 1. Decision Regions for an N = 8 Option PSIss

CODE OPTION	F_0 REGION IN BITS
PSI1,0	$F_0 \le 5J/2$
PSI1,1	$5J/2 < F_0 \le 9J/2$
PSI1,2	$9J/2 < F_0 \le 17J/2$
PSI1,3	$17J/2 < F_0 \le 33J/2$
PSI1,4	$33J/2 < F_0 \le 65J/2$
PSI1,5	$65J/2 < F_0 \le 129J/2$
PSI1,6	$129J/2 < F_0 \le (128n\text{-}831)J/2$
PSIbu	$(128n\text{-}831)J/2 < F_0$

PSIss Implementation Parameters

The primary PSIss parameters used in the first-generation VLSI implementations were as follows:[6-9]

a) J = block size of $\tilde{\delta}^n$ = 16

b) n = supported quantization of $\tilde{\delta}^n$ samples
= 12 for JPL chips
= 4, 5, ..., 14 for MRC chips

c) N = number of code options
= 11 for JPL chip
= 12 for MRC chip

d) λ = Dynamic Range Parameter = 1 = Starting Option Parameter

Some Subtle Differences. By Eq. 23, the number of fixed identifier bits does not need to be more than $[\log_2 N]$ bits. A coder which represents 8-bit data has no need for more than 8 code options and thus should need no more than a 3-bit identifier. The second-generation MRC design will recognize this situation. However, JPL's first- and second-generation chips and MRC's first generation chip fixed the number of identifier bits at four.

In the JPL case, the first-generation chip only supported input quantization of 12 bits/sample, so this was not an issue. A second design now provides support for data with quantization down to 8 bits/sample. But in doing so the number of code options is still fixed at 11, yielding 4 bits for identifiers.

The JPL support for $n < 12$ is provided by treating data of lesser quantization as right-justified 12-bit data. The consequence of this is that there does not exist a true backup (default) mode when input data is not truly 12 bits/sample. For example, instead of using an 8 bits/sample PSIbu (e.g., as dictated by Table 1), a JPL second generation coder would instead use some intermediate split-sample mode beyond PSI1,6. In the rare eventuality that PSIbu is needed, this shortcoming would incur a penalty of over 1 bit/sample in the block in which it occurs. Each MRC design uses a true backup.

By (22) and (24), PSIbu in (7) is always assigned the identifier id = N−1. Thus an N=11 option coder would assign the binary four-tuple identifier "1010" (for ten).

JPL's N=11 designs instead assign the "all-ones four-tuple" to the PSIbu identifier. Similarly, and more generally, the MRC designs assign the "all-ones four tuple" to the PSIbu identifier for N>8 and the "all-ones three-tuple" when $N \leq 8$.

Another distinction between the JPL and MRC designs lies in the method for determining which code option to use. The JPL coders basically use the approach illustrated in Table 1 where decisions are based on F_0 alone. (This is generally the most desirable technique for software implementations because of the minimal computation requirements.) MRC coders instead make decisions based on the exact bit count for each option (requiring the calculation of each F_k in (28)). The difference in average performance between the two methods has been shown to be of no practical significance.

By (21), (13) and (14), the form of a coded block is

$$ID(id) * PSI1(\tilde{M}^{n,k}) * \tilde{L}_k$$

Here, the JPL and MRC approaches diverge slightly. The MRC format follows the definition of $\tilde{L}_k$ given by (11) and (12). However, the JPL format splits $\tilde{L}_k$ further into k subsequences, each containing all the sbs from each sample which are of the same significance. While this provided a simplicity in the JPL coder design, it incurs a penalty on the operations required by a decoder.

Some additional differences are noted in a later section on pre-processing. For specific details on these implementations, consult Refs. 6–16.

More Generality

The adaptive coder we have designated as PSIss is actually a subset of the more general coders in Ref. 15 (PSI14 and PSI14,K). The latter coder definitions

1) Permit λ to be zero or negative, thus allowing for additional "low-entropy" code options that provide improved performance below 1.5 bits/sample. Ref. 15 and earlier papers provide various low-entropy code options which can be incorporated into this structure or can stand alone as separate approaches. Yeh has provided a

computationally simple algorithm for incorporation in the second-generation MRC design.[14]

2) Provide a second parameter, K, for adjusting the Dynamic Range. K basically specifies a fixed K-bit pre-split of data samples, before or after a pre-processor.

3) An extended coding structure that allows these same algorithms to also be applied to the representation of code identifiers. This reduces the overhead penalty when operating many-optioned coders on fairly stationary data.

IV. THE PRE-PROCESSOR

The entropy coder, PSIss, as described in the previous section, was designed to efficiently represent (pre-processed) Standard Data Sources. PSIss doesn't need to know which pre-processor was used to produce its input. However, for an extremely broad set of real problems, the general pre-processing function of Fig. 1 can be replaced by the more specific Basic Predictive Pre-processor in Fig. 4. It is shown imbedded within the complete coding module for your convenience.

Standard Predictor

The first part of this pre-processor is a very simple predictor consisting of a single sample-delay element. With x_i as the ith sample in $\tilde{X}^n$, this delay element "predicts" that x_i equals the previous sample:

$$\hat{x}_i = x_{i-1} \tag{35}$$

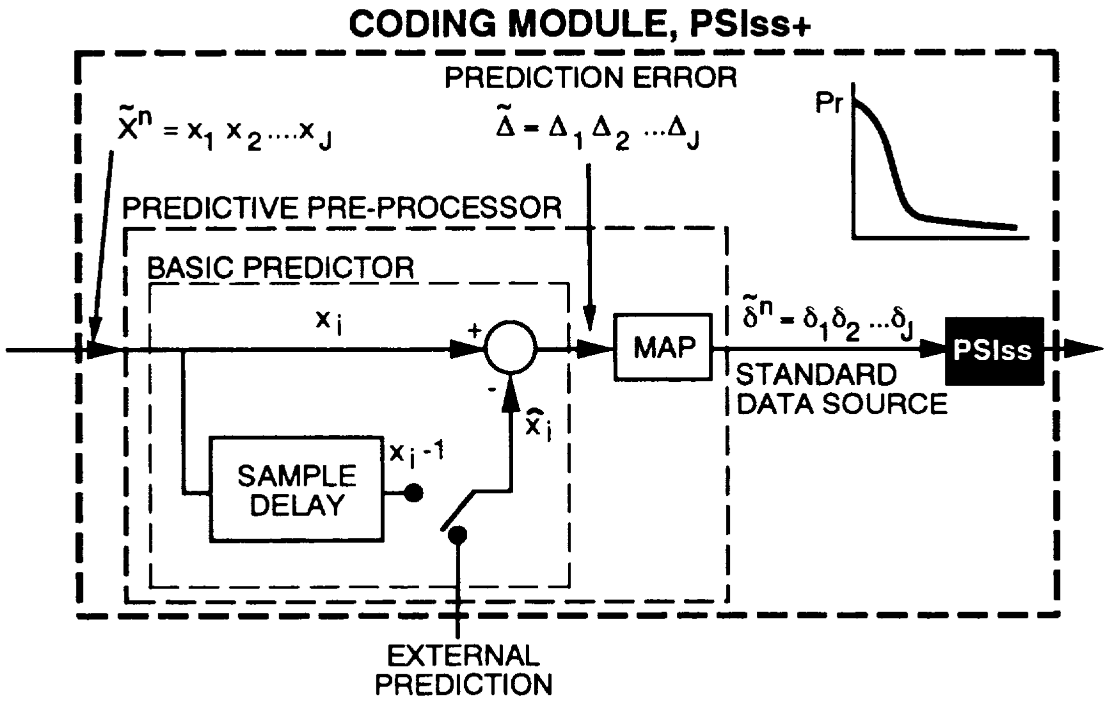


Fig. 4. **Basic Predictive Pre-processor Within Coding Module, PSIss+**

The previous sample could be the last sample from a previous block when coding multiple blocks. It is assumed that the sample delay is always initialized with a prediction. But note also at this time that the module's design allows for an External Prediction to be supplied in place of this simple one-dimensional form.

The difference between any sample and its prediction produces the error signal

$$\Delta_i = x_i - \hat{x}_i \tag{36}$$

and the block of J error values

$$\tilde{\Delta} = \Delta_1 \Delta_2 \ldots \Delta_J \tag{37}$$

As a new data source, $\tilde{\Delta}$ sequences tend to be uncorrelated and display a unimodal distribution around zero (when data values are not near the boundaries of its dynamic range). That is[4]

$$\Pr[\Delta_i = 0] \geq \Pr[\Delta_i = -1] \geq \Pr[\Delta_i = +1] \geq \Pr[\Delta_i = -2] \geq \ldots . \tag{38}$$

The Mapper (into Standard Source)

When the latter condition in (36) is true, the following function will map each Δ_i into a corresponding Standard Source δ_i such that

$$p_0 \geq p_1 \geq p_2 \geq p_3 \geq \ldots .$$

Additionally, it will assure that an n-bit/sample x_i from $\tilde{X}^n$ produces an n-bit/sample δ_i. Further, the desired probability ordering of the δ_i is more closely approximated when x_i values are near 0 or $x_{max} = 2^n - 1$.

---

[4] An equally valid assumption is

$$\Pr[\Delta_i = 0] \geq \Pr[\Delta_i = +1] \geq \Pr[\Delta_i = -1] \geq \Pr[\Delta_i = +2] \geq \ldots$$

which is the basis of the JPL VLSI chips. It leads to a mapping function which looks very similar to (39) and (40). Reference 15 shows that a "decoder" using one mapping function could be used to decode data that had been generated using the other mapping function.

The Mapper, used in the MRC design, is defined by

$$\delta_i = \begin{cases} 2\Delta_i & 0 \le \Delta_i \le \theta \\ 2\,|\Delta_i| - 1 & -\theta \le \Delta_i < 0 \\ \theta + |\Delta_i| & \text{Otherwise} \end{cases} \tag{39}$$

where

$$\theta = \min(\hat{x}_i\,,\, x_{max} - \hat{x}_i)$$
$$x_{max} = 2^n - 1 \tag{40}$$

A Further Benefit. Suppose the pre-processor in Fig. 4 is set to receive an external prediction, and **fix** that prediction to

$$\hat{x}_i = 0 \tag{41}$$

Then, tracing through (39) and (40), one finds that

$$\delta_i = x_i \tag{42}$$

That is, the input to the pre-processor, $\tilde{X}$, is passed directly through unchanged, becoming as $\tilde{\delta}$. This means that the separate desired external input line in Fig. 1 (to allow the pre-processor to be skipped) can be omitted.

The Ideal Case

If one assumes that the distribution of $\tilde{\Delta}$ samples in (38) fits the Laplacian form, then the code equivalence result in (16) can be proven.[17,18] That is,

the simple PSI1,k codes are equivalent
to Huffman Codes for Laplacian
distributions of prediction errors. (43)

Reference Sample

Most of those applications which make use of the built-in Predictive Preprocessor will occasionally need to incorporate a "Reference Sample" along with the coded prediction errors (e.g., at the start of an image line). Each of the VLSI implementations incorporate an optional feature to extract such a Reference Sample from an incoming data stream. The JPL and MRC approaches to formatting this Reference Sample are distinctly different. Consult Refs. 10, 11, 14 and 15.

V. PERFORMANCE COMPARISONS

References 12 and 18 compare the performance of a complete PSIss+ coding module with the well-known Lempel-Ziv, Adaptive Huffman and Arithmetic coding algorithms.

References

1) R. F. Rice and J. R. Plaunt, "Adaptive Variable Length Coding for Efficient Compression of Spacecraft Television Data," **IEEE Trans. on Communication Technology**, Vol. COM-19, Part I, Dec. 1971, pp. 889-897.

2) R. F. Rice, "Some Practical Universal Noiseless Coding Techniques," **JPL Publication 79-22**, Jet Propulsion Laboratory, Pasadena, California, March 15, 1979.

3) R. F. Rice, "Practical Universal Noiseless Coding," **1979 SPIE Symposium Proceedings**, Vol. 207, San Diego, California, August 1979.

4) R. F. Rice and J. Lee, "Some Practical Universal Noiseless Coding Techniques," Part II, **JPL Publication 83-17**, Jet Propulsion Laboratory, Pasadena, California, March 1983.

5) R. F. Rice, "Universal Noiseless Coding Techniques (Electronic Slideshows)," **NASA Workshop on Data Compression**, Snowbird, Utah, May 1988.

6) J. Lee et al., "A Very High Speed Noiseless Data Compression Chip for Space Imaging Applications," **Proceedings IEEE Data Compression Conference,** Snowbird, Utah, April 1991.

7) J. Venbrux and N. Liu, "Lossless Image Compression Chip Set," **Proceedings of Northcon**, Seattle, Washington, Oct. 9-11, 1990, pp. 145-150.

8) J. Venbrux and N. Liu, "A Very High Speed Lossless Compression/ Decompression Chip Set," **JPL Publication 91-13**, Jet Propulsion Laboratory, Pasadena, California, July 15, 1991.

9) J. Venbrux and Pen-Shu Yeh, "A VLSI Chip Set for High-Speed Lossless Data Compression," **IEEE Transactions on Circuits and Systems for Video Technology**, Vol. 2, No. 4, December 1992.

10) ------------, "Specification for first generation MRC lossless coder chip set," **Advanced Hardware Architectures, Inc.**, Moscow, Idaho, 1991.

11) J. Bowers, "Chip Specification for the Cassini Data Compression Chip," **JPL Internal Memorandum,** Jet Propulsion Laboratory, Pasadena, California, December 1, 1992.

12) J. Venbrux et al., "VLSI Chip Set Development for Lossless Data Compression," **Proceedings of the AIAA Computing in Aerospace 9 Conference**, San Diego, California, October 19–21, 1993.

13) J. Miko et al., "A High Speed Lossless Data Compression System for Space Applications," **Proceedings of the AIAA Computing in Aerospace 9 Conference,** San Diego, California.

14) J. Venbrux, "Universal Source Encoder for Space (USES) Specification," **Internal Document, Microelectronics Research Laboratory,** University of New Mexico, Albuquerque, New Mexico.

15) R. F. Rice, "Practical Universal Noiseless Coding Techniques," Part III, **JPL Publication 91-3**, Jet Propulsion Laboratory, Pasadena, California, November 15, 1991.

16) R. F. Rice et al., "Algorithms for a Very High Speed Universal Noiseless Coding Module," **JPL Publication 91-1**, Jet Propulsion Laboratory, Pasadena, California, February 15, 1991.

17) Pen-Shu Yeh et al., "On the Optimality of Code Options for a Universal Noiseless Coder," **JPL Publication 91-2**, Jet Propulsion Laboratory, Pasadena, California, February 15, 1991.

18) Pen-Shu Yeh et al., "On the Optimality of a Universal Lossless Coder," **Proceedings of the AIAA Computing in Aerospace 9 Conference**, San Diego, California, October 19-21, 1993.

APPENDIX B

A Practical Concept for Improving the MSTI Downlink Data Rate by 2:1 to 3:1

OVERVIEW/SUMMARY

Thanks to Ray Rew of Spectrum Astro, new light has been shed on the characteristics and elements of the now and future MSTI downlink.[1] Based on initial discussions with Ray, it would appear very likely that there is a technically and financially practical approach to upgrading the post-MSTI 3 downlink data rate capabilities by a factor of 2 to 3.

Basically, the approach would require minor modifications to the current modulation system and the incorporation Reed-Solomon coding and decoding chips within the communication system.

The inclusion of Reed-Solomon (RS) coding within a BMDO program would itself provide a second demonstration of modern communications technology (in addition to data compression). It would offer substantial long-term benefits to future programs in addition to the immediate 2-3:1 downlink improvement. (4:1 to 6:1 overall when combined with image compression.)

This brief report seeks to provide:

a) Historical perspective of RS coding;
b) Performance characteristics associated with the MSTI data link;
c) Technology availability.

In the interest of securing the potential advantages of RS coding at the earliest possible date (e.g., MSTI 4 application) it is important that the information provided here be considered and acted on as soon as possible.

THE UNCODED DOWNLINK

Error Vulnerability

The effect of individual communication bit errors on uncompressed data is generally isolated to the data sample in which they reside. But the effect on compressed data can be far more devastating. Basically, a single error could cause a decompressor to become confused until it can be re-initialized (e.g., possibly on a line-by-line basis for the MSTI 3 image compression demonstration).

A similar error susceptibility is associated with the encryption techniques which are currently planned. In both cases, single errors tend to cause "bursts" of errors (this will be looked at more closely later).

One can avoid the consequences of this error vulnerability by assuring that the frequency of errors is reduced to an acceptably low value. At this time, acceptably low has come to mean a bit-error probability for MSTI of [1]

$$P_b < 10^{-6} \quad (1)$$

The Uncoded Channel

The MSTI 2 communication system (and presumably MSTI 3 and on) employ an "uncoded" binary data link[1] in the sense that it includes no "error-correcting" coding. In the absence of bandwidth constraints, which we will assume for now, the MSTI uncoded data link looks exactly like a similarly named uncoded, deep-space data link in terms of performance characteristics. Channel error characteristics are shown in Fig. 7, which plots "bit-error" probability, P_b , versus signal-to-noise ratio, E_b/N_o (energy-per-bit) in decibels (dB).

E_b/N_o vs. Rate. Assuming that transmission power, receiver/transmitter antennas (and their proximity) remain fixed, E_b/N_o can be viewed as the inverse of data rate. Increasing the data rate by a factor of two (3 dB) doubles the E_b/N_o , and vice versa. (The energy available per unit of time applies to a bit's signal twice as long.)

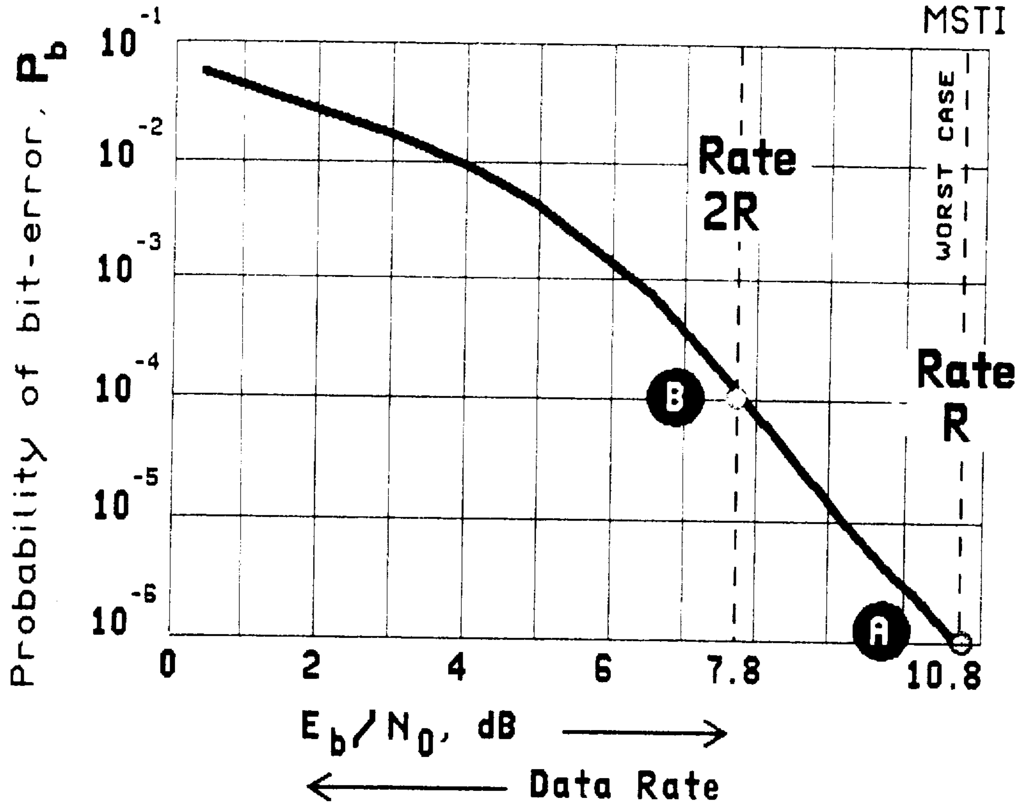


Fig. 1. The Uncoded Performance Curve

Point A in Fig. 1 indicates the point on this curve ($E_b/N_o \approx 10.8$ dB, rate = R bits/s) where the frequency of "random" bit errors is 10^{-6}. This is currently the worst case operating point for MSTI as specified in (1). That is, there will usually be a greater E_b/N_o than required to achieve a $P_b = 10^{-6}$.[1] This is called Margin, and assures that throughout a mission the conditions in (1) are met.

Point B on the Uncoded Link Performance curve indicates where $P_b = 10^{-4}$. This occurs roughly 3 dB lower where [$E_b/N_o \approx 7.8$ dB, rate $\approx$ 2R bits/s] . Thus

a) If the symbol (bit) rate over the channel could be increased (the bandwidth question)

and

b) If it would be possible to accept an error rate as high as $P_b = 10^{-4}$

then, an increase in data rate by a factor of two could be achieved (with the same margin).

Bandwidth. In the discussions with Ray Rew[1] in March, he indicated that some minor modifications to the current modulation system should allow an increase in channel symbol rate by 3-4 dB. (2) (Of course, Ray would need to more clearly indicate what "minor" means.)

This would seem to take care of item (a) above (although we will later be interested in transmitting channel symbols at rates higher than 3R). However, the error vulnerability of both compressed and encrypted data makes condition (b) unacceptable.

CODING THE CHANNEL

Similar situations existed in the deep-space program in the 1970's. Before the first inclusion of image data compression on Voyager at Uranus, the accepted error rate for uncompressed image data was in the vicinity of 10^{-3}. To include error-sensitive data compression with the communication channels of the time would have required a reduction in data rate by approximately a factor of two. This is basically the situation we have here with MSTI, except that we've already reduced the data rate (to get to $P_b = 10^{-6}$).

The deep-space situation was somewhat more complicated than the present MSTI problem. The baseline communication system for Voyager (and other projects at the time) employed a convolutionally encoded/Viterbi decoded channel which would operate at bit error rates as high as 10^{-3}. This was acceptable for pre-data-compression imaging data but not for other forms of data (which generally constituted a small fraction of the total data). Hence, the non-imaging data was additionally coded with a (24,12) Golay block code (before reaching the convolutional coder).

AICS[2]-[4]

The Advanced Imaging Communication System (AICS) was introduced as a solution in the mid 1970's.

AICS employed a powerful, interleaved Reed-Solomon code in combination with the convolutional channel. All data, compressed or not, could be communicated through this concatenated channel "virtually error free" at roughly the same data rate at which the convolutional channel alone achieved a bit-error probability of only 10^{-3}.

AICS was patented by NASA in 1976 and the concatenated RS/Viterbi channel has become a NASA and International Standard. It appears on virtually all deep space missions, starting with Voyager (at Uranus). Various developments have produced both RS coder and decoder chips with essentially the same code parameters as originally specified in Ref. 2.

On the surface, these developments would seem to have little to do with the MSTI problem, since MSTI currently employs an uncoded link. On the contrary, we just need to look at the problem a little differently.

References 2-4 provide the original arguments and technical details that were originally needed to convince doubters of AICS performance and feasibility (data compression had not yet been used either). Subsequent sections will extract portions of these references, and others, to address the MSTI application.

The Reed-Solomon Code

A simple block diagram of a Reed-Solomon coder with parameters J and E is shown in Fig. 2.

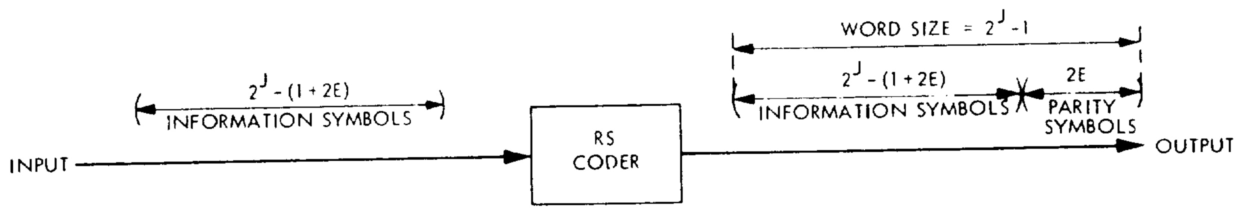


Fig. 2. RS Coder

Such an RS code is non-binary. An RS symbol consists of J bits so there are 2^J possible RS symbols. All coding and decoding operations involve RS symbols, not individual bits.

The input to an RS Coder are called "information symbols." As Fig. 2 shows, $2^J-(1+2E)$ information symbols enter the RS Coder to the left. The result of coding is a codeword of length 2^J-1 symbols of which the first $2^J-(1+2E)$ are the same symbols entering on the left. This makes the code systematic. The remainder of the codeword is filled in with 2E parity (check) symbols.

An RS symbol is in error if any of the J bits making up the symbol are in error. E represents the number of correctable RS symbol errors in an RS codeword. That is, if E or less RS symbols are in error in any way, the coder will be capable of correcting them.

Tying Down Parameters. For a great many reasons the code parameters chosen originally for AICS and eventually implemented in VLSI are

$$J = 8 \quad , \quad E = 16 \tag{3}$$

Thus RS symbols are bytes and a decoder can correct up to 16 "symbol" errors in a single codeword. The RS code structure with these parameters is shown in Fig. 3.

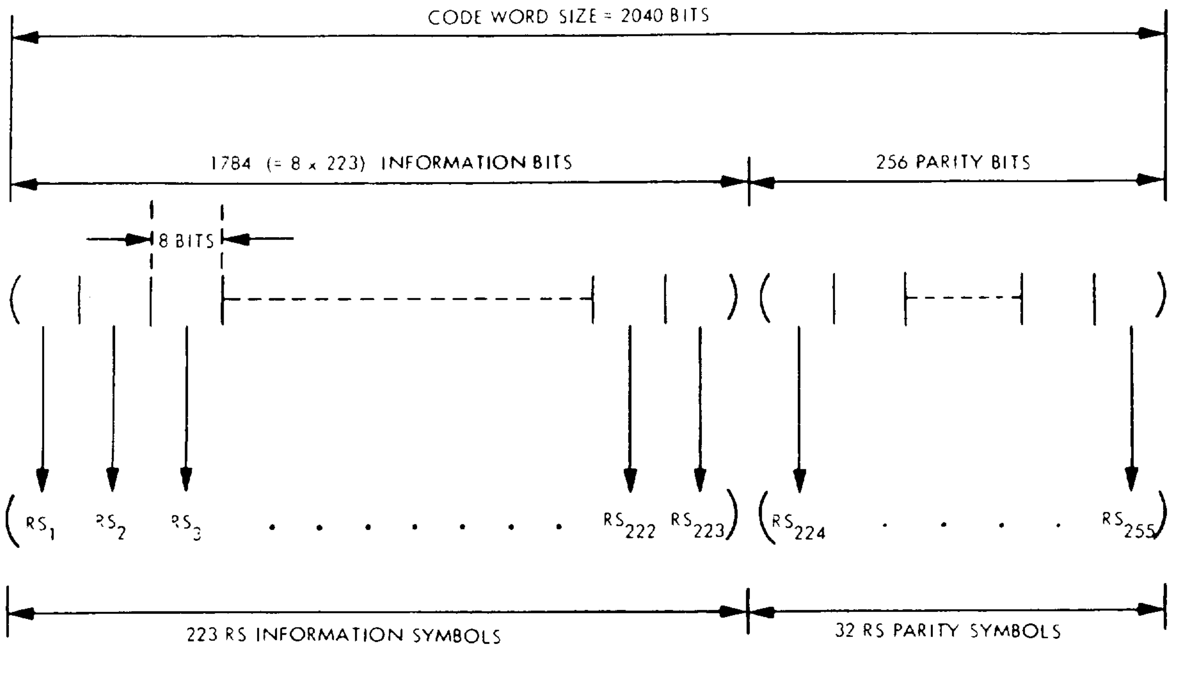

CODE WORD SIZE = 2040 BITS
1784 (= 8 x 223) INFORMATION BITS
256 PARITY BITS
8 BITS
RS1 RS2 RS3 . . . . RS222 RS223
RS224 . . . . . RS255
223 RS INFORMATION SYMBOLS
32 RS PARITY SYMBOLS

Fig. 3. Basic RS Codeword Structure, J = 8 , E = 16

Interleaving. Errors made by a Viterbi decoded convolutional channel tend to be "bursty". That is, errors occur together in clumps (increasingly so at higher error rates). The symbol error correcting capabilities of this RS code is ideally suited and matched to these characteristics. However, such "burstiness" becomes increasingly severe at high (convolutional channel error rates). To avoid a loss in performance, deep space RS coding systems employ "interleaving" to spread the effect of long inner-channel error bursts over several codewords.

As understood now, a near-term MSTI application of RS coding is unlikely to require the deep-space interleaving capability because an uncoded data link does not inherently have any of these "bursty" characteristics. The currently available RS coder/decoder chips can set interleave depth, I, from I = 1 (none), up to I = 8.

While not immediately an issue, there are certainly many military (SDI) situations in which error events come in bursts, even when the underlying channel is uncoded (e.g., other components of the end-to-end data link). Thus, for future reference, it is important to at least understand the potential burst-error-correcting capability of an interleave RS code.

The basic burst-error-correcting capability of this J = 8, E = 16 code is a burst of E = 16 symbols or 128 bits. That is, a complete wipe out of any stretch of 128 bits in a codeword (without other errors) could be corrected.

Interleaving at a depth I basically increases the burst error correcting ability by the factor I. <u>Then with I = 8, an error burst of over 1000 bits could be corrected</u> (assuming no other errors in the 8 codewords).

RS Coding on Uncoded Channel

The performance curve for the uncoded data link of Fig. 1 appears again in Fig. 4 along with the performance curve for an RS coder with the just specified parameters J = 8, E = 16, $I \geq 1$.[5]

As in Fig. 1, error probability is plotted as a function of E_b/N_o in Db, which we know is inversely related to data rate. Points A and B are repeated and their relative data rates are indicated as R bits/s and 2R bits/s, respectively.

An RS performance curve is shown as the new graph. Here the ordinate is "probability of a Reed-Solomon codeword error," P_{RS} , instead of bit-error probability (which would be lower). This is a much more appropriate measure of performance as we will illustrate later.

Note that "signal-to-noise" ratio needs a more sophisticated interpretation here. E_b/N_o means energy per <u>information</u> bit. On an uncoded data link the information bits are the same as channel bits so this issue never comes up. But with RS coding, not all the bits communicated are information bits. Roughly 12% are parity symbols with the J = 8, E = 16 code.

The RS performance curve shown takes this into account and reflects the true information rate (the data rate of the real data). Channel bits are actually communicated at about 0.6 dB lower signal-to-noise ratio (and 0.6 dB higher bit-rate).

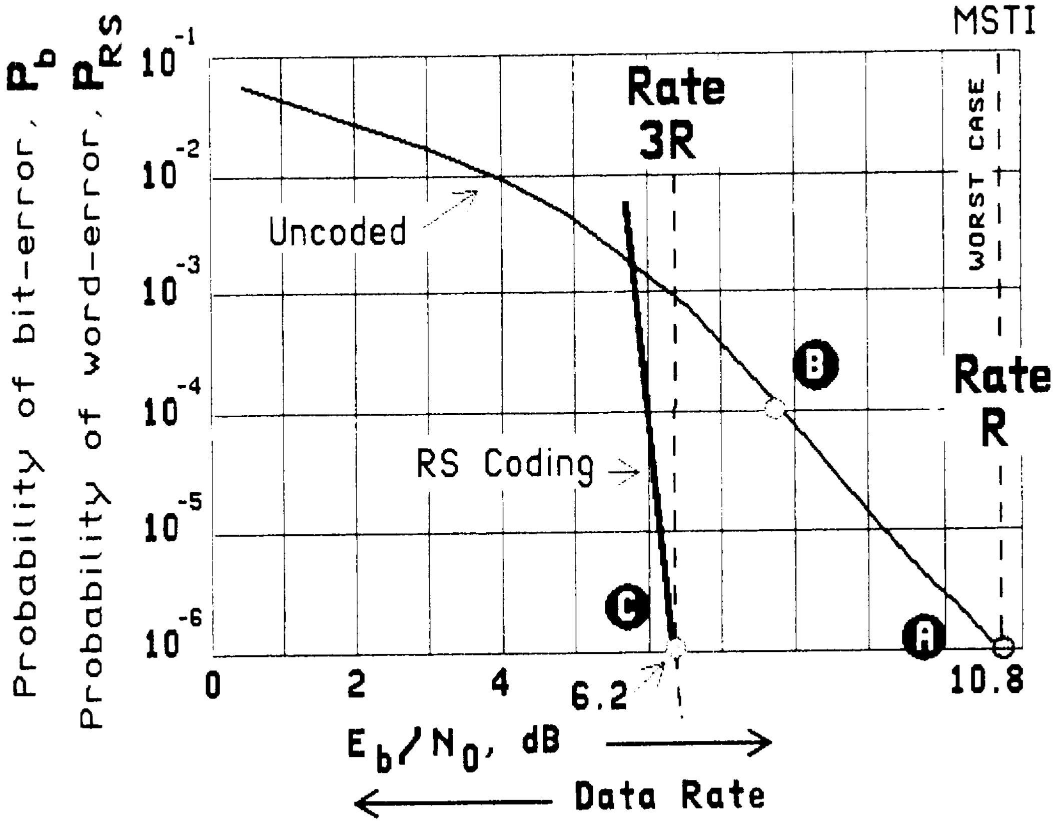


Fig. 4. RS Performance Curve

As should be apparent, the P_{RS} performance curve is quite different than that for the uncoded Data Link. Not only is it located at a lower E_b/N_o (and hence higher data rate), but it is very steep. Error rates drop off by approximately one order of magnitude for every 0.1 dB increase in E_b/N_o. Basically, once the E_b/N_o exceeds around 6.2 dB (point C), where $P_{RS} < 10^{-6}$, a "virtually error-free" data link is achieved. The true meaning of a $P_{RS} < 10^{-6}$ is discussed in a later section titled "Error Events."

Data Rate Improvement. The difference in E_b/N_o, and hence data rate, between Point A and Point C in Fig. 4 provides the potential data rate improvement to post-MSTI 3 missions from a direct application of RS coding. The potential data rate improvement is

$$\Delta \text{ Rate} \approx 4.6 \text{ dB} \tag{4}$$

which is equivalent to a factor of 3. This is illustrated in Fig. 4 where Point A is indicated to be at data rate, R, and Point C at data rate 3R.

Actual improvements will depend on what modifications to the current modulation system are feasible. Such modifications need to allow a higher (channel) symbol rate without bandwidth constraints. Initial conversations with Ray Rew suggest that a factor of two or so should be quite feasible.[1]

Concatenated RS Coding

It is of longer-term SDI interest to understand the potential for data rate improvements based solely on the available E_b/N_o . That is, when modulation systems can accommodate the higher bandwidth. Unlike the direct RS coding application, the following discussion is not expected to be of immediate interest to the MSTI program.

The same RS code, concatenated with a $K = 7$, $v = 2$ convolutional code (Voyager) could operate at approximately 3.2 dB E_b/N_o to achieve the equivalent of Point C in Fig. 4 That's another factor of two - for a total data rate improvement of 6.

Adding Compression. Coupled with an assumption of a 2:1 lossless compression capability, this yields a total net potential of 12:1 improvement in downlink capability from advanced technology - without any loss to the data.

If it were later determined that frame-to-frame prediction could improve the lossless compression factor by up to 6:1 for some images - then, on those images, the net gain through the downlink would approach 36:1.

Bandwidth Expansion. To operate the $K = 7$, $v = 2$ codes at such low E_b/N_o would require a channel symbol rate which is a factor of 4 higher than required of the uncoded data link to achieve Point C (i.e., 12:1+ compared to Point A). Some other less powerful codes (e.g., with $v = 4/3$) could get close to these results with a channel symbol rate increase closer to 3:1.

Going the other way, a further data rate improvement of about 1 dB could be achieved with more powerful convolutional codes (e.g., $K = 15$, $v = 4$) and greater bandwidth expansion.

Error Events and P_{RS}

An error event on the uncoded channel occurs whenever an individual bit error occurs. Such independently occurring random errors will tend to be distributed uniformly throughout a data stream.

An error event on a channel which incorporates the RS coding occurs only when an RS codeword is in error (i.e., when more than 16 symbol errors occur). Since a codeword is roughly 2000 bits in length, this means that, with a codeword error rate of P_{RS} , an error event will occur once every

$$\frac{2\text{x}10^3}{P_{RS}} \qquad (5)$$

Taking $P_{RS} = 10^{-6}$ as assumed for Point C in Fig. 4, this would mean <u>one error event every</u>

$$2 \times 10^3 \times 10^6 = 2 \times 10^9 \text{ bits} \tag{6}$$

Accepting a decrease in data rate by a tiny 0.1 dB (i.e., at 3.3 dB) would increase this figure to <u>one error event every</u>

$$20 \text{ Gbits} \tag{7}$$

or close to <u>one error event every</u>

$$3 \text{ Gbytes} \tag{8}$$

Surely, the term "virtually error free" is appropriate for this kind of performance, and suggests the possibility of an error-free mission.

Effect of Error Events. A decoded RS symbol stream will remain entirely error-free unless a codeword error occurs. When a codeword is in error, more than 16 symbol (byte) errors have occurred and so they are not correctable by the 16 error-correcting code decoder. (In most cases, there will be 17 symbol errors remaining). Such symbol errors will be randomly distributed throughout a codeword, including its non-information parity symbols. We wish to consider the worst case impact of such errors on compressed image data.

Figure 5 provides a somewhat idealized illustration which should be adequate for this purpose. Here a sensor produces source image lines which are 512 pixels long, with each pixel quantized to 8 bits/pixel. Images are delineated by parentheses, '(' and ')' in this figure. The central three source lines in Fig. 5 are labelled L1, L2 and L3.

A source line is assumed to be prefaced by a synchronization word (SYNC), of length ε bits. The total length of an uncompressed source line is then $4096 + \varepsilon$ bits, where $\varepsilon << 4096$.

Now assume, somewhat idealistically, that the data part of each line is compressed by 2:1 (e.g., using a lossless Rice Algorithm system as currently envisioned) and that each compressed line is preceded by a synchronization word (SYNC) and a Reference Sample (REF), of combined length ε'. The length of a compressed line is then $2048 + \varepsilon'$ bits, where $\varepsilon' << 2048$. Uncompressed lines L1, L2 and L3 in Fig. 5 become compressed lines cL_1, cL_2 and cL_3.

If an error is encountered by a decompressor it will very likely cause the effect of the error to propagate until the decompressor can be re-initialized. The SYNC and REF assumptions provide the assurance that, with very high probability, a decompressor can be reset at the start of each line. Thus, the impact of "an error" can be confined to a line. Then <u>a worst case assumption for the impact of a single error is that it will cause the complete loss of the line it occurs in</u> (i.e., as if the error occurred at the start of the line).

Under this assumption of worst case damage from a single error, <u>multiple errors occurring within a line are no more damaging than one</u>. This bounds the problem in a practical way.

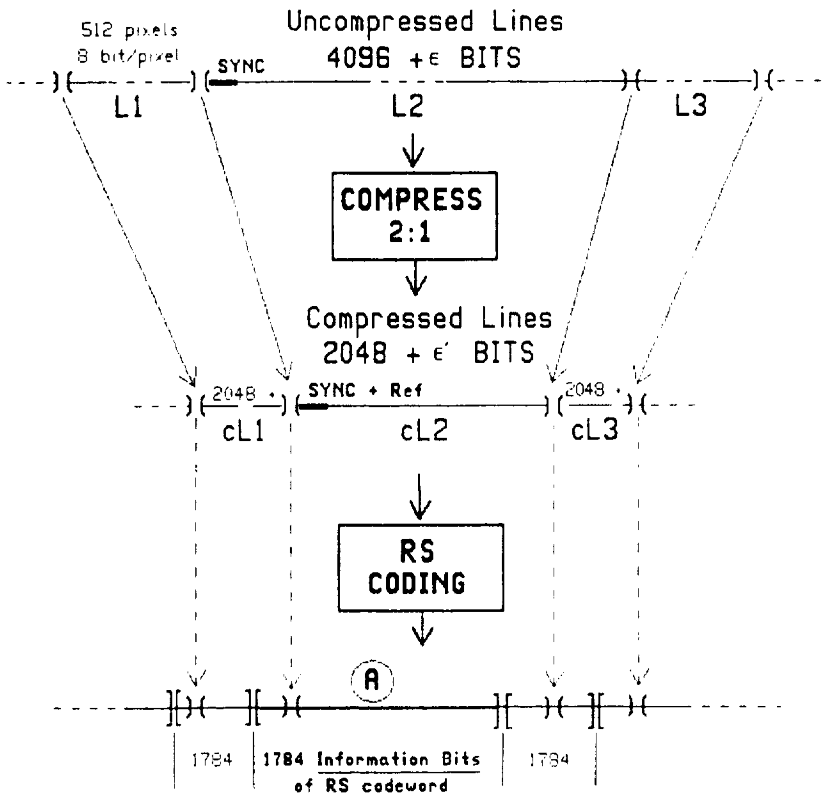


Fig. 5. Error Effects on Compressed Lines

Now consider Fig. 5 again. The lower part of the diagram assumes that the compressed lines are passed through a J = 8, E = 16 RS coder without interleaving (I = 1). Generally, the start of a line and the start of an RS codeword will not line up as Fig. 5 illustrates. Here the square brackets, '[' and ']' , are used to show the start and end boundaries of the 1784 Information Bits of sequential RS codewords.

Suppose that the codeword marked A in Fig. 5 is in error. Extending our worst case criteria, compressed lines with any portions that lie within this RS codeword are lost. In this example, lines L1 and L2 would be lost (although we can easily see that only the end of L1 would be damaged).

Summary Observations. Combining these observations with those in (8) we have, for a system incorporating basic lossless image compression:

> Operating the RS coded link at an $E_b/N_o > 3.3$ dB (corresponding to an improvement factor in data rate of $\approx$ 3:1 over the uncoded data link), would incur a worst case loss of only two lines of image data once every 6 Gbytes of source data! (That is, once per mission?) (9)

By comparison,

> An uncoded data link operating at a bit-error probability of $P_b = 10^{-6}$ (i.e., at an $E_b/N_o \approx 10.8$ dB, or a data rate only 1/3 of the proposed RS coded system), would incur a worst case expected loss of 24,000 lines of image data for the same 6 Gbytes source data. (10)

Effect of Error Events II. Some additional observations can be made.

Interleaving

Interleaving might be incorporated to extend the burst-error correcting ability of the RS coding and/or to avoid (E_b/N_o) performance degradation when operating in combination with a convolutional inner channel.

The built-in interleaving approach of available chips is called Interleave B in Ref. 2. In this case, increasing interleave depth, I , will spread the errors of a single codeword over additional compressed lines. Specifically, errors may occur over a stretch of 1784I bits, instead of only 1784 bits.

Performing the interleave A approach in Ref. 2 would assure that the effect of a single codeword error remained confined to only 1784 data bits regardless of interleave depth, I .

More Sophisticated Compression

If the lossless compression system now being investigated, and illustrated in Fig. 5, were extended to include two-dimensional prediction (to improve performance by ≈ 0.5 bits/pixel), individual lines would no longer be independent. The effect of an error could propagate over multiple lines (instead of only to the end of one). This problem would be further compounded if prediction between image frames were employed. Errors could, in this case, impact multiple frames. Similar situations can be associated with the application of "lossy" algorithms too.*

Certainly, the advantages of the virtually error-free environment provided by Reed-Solomon coding to these situations is obvious.

TECHNOLOGY AVAILABILITY

Reed-Solomon coder and decoder chips with the desired $J = 8$, $E = 16$ parameters have been developed by the University of New Mexico under the direction of GSFC.[6] The CMOS coder chips are space qualified.

- Radiation-hardened to 3 x 10^6 rads total dose
- SEU-hardened to less than 1 x 10^{-10} error per bit-day
- Pin selectable interleaving, I = 1-8
- 200 Mbits/s capability

COST[7]

Decoders. $2K per 4-chip decoder.

Flight Coders. Minimum order of 25 chips at $1200 per unit, plus $7K testing per lot, for a total of ≈ $37K.

*Current de-encryption algorithms have a similar difficulty with errors as compressed data. Many errors are generated from a single channel error.

REFERENCES

1. Technical Interchange with Ray Rew of Spectrum Astro at the Jet Propulsion Laboratory, Pasadena, CA, March 23, 1993.

2. R. F. Rice, "Channel Coding and Data Compression System Considerations for Efficient Communication of Planetary Imaging Data," *JPL Technical Memorandum 33-695*, Jet Propulsion Laboratory, Pasadena, CA, June 1974.

3. R. F. Rice, "An Advanced Imaging Communication System for Planetary Exploration," *SPIE Seminar Proceedings, Vol 66*, August 21-22, 1975, pp. 70-89.

4. R. F. Rice, "End-to-End Imaging Information Rate Advantages of Various Alternative Communication Systems," *JPL Publication 82-61*, Jet Propulsion Laboratory, Pasadena, CA, September 1, 1982.

5. CCSDS Report *CCSDS 700.0-G-2*, October 1989, pg. A-5.

6. Data Sheet for "Space Qualified High Speed Reed-Solomon Encoder," NASA SERC for VLSI Systems Design, University of New Mexico, Albuquerque, NM, September 1992.

7. Conversation with Warner Miller, GSFC, April 6, 1993.

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. 94-16	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle System Considerations for Efficient Communication and Storage of MSTI Image Data		5. Report Date October 1994
		6. Performing Organization Code
7. Author(s) Robert F. Rice		8. Performing Organization Report No.
9. Performing Organization Name and Address JET PROPULSION LABORATORY California Institute of Technology 4800 Oak Grove Drive Pasadena, California 91109		10. Work Unit No.
		11. Contract or Grant No. NAS7-1260
		13. Type of Report and Period Covered JPL Publication
12. Sponsoring Agency Name and Address NATIONAL AERONAUTICS AND SPACE ADMINISTRATION Washington, D.C. 20546		14. Sponsoring Agency Code RF 184 BP-889-54-49-10-20
15. Supplementary Notes		

16. Abstract

The Ballistic Missile Defense Organization has been developing the capability to evaluate one or more high-rate sensor/hardware combinations by incorporating them as payloads on a series of Miniature Seeker Technology Insertion (MSTI) flights. This publication represents the final report of a 1993 study to analyze the potential impact of data compression and of related communication system technologies on post-MSTI 3 flights.

Lossless compression is considered alone and in conjunction with various spatial editing modes. Additionally, JPEG and Fractal algorithms are examined in order to bound the potential gains from the use of lossy compression. But lossless compression is clearly shown to better fit the goals of the MSTI investigations. Lossless compression factors of between 2:1 and 6:1 would provide significant benefits to both on-board mass memory and the downlink. For on-board mass memory, the savings could range from $5 million to $9 million. Such benefits should be possible by direct application of recently developed NASA VLSI microcircuits.

It is shown that further downlink enhancements of 2:1 to 3:1 should be feasible through use of practical modifications to the existing modulation system and incorporation of Reed-Solomon channel coding. The latter enhancement could also be achieved by applying recently developed VLSI microcircuits.

17. Key Words (Selected by Author(s)) 1. Aircraft communications and navigation 2. Spacecraft communications, command, and tracking 3. Communications 4. Systems analysis		18. Distribution Statement Unclassified; unlimited	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 82	22. Price

JPL 0184 R 9/83