PROPOSED DATA COMPRESSION SCHEMES FOR THE GALILEO S-BAND CONTINGENCY MISSION *

Kar-Ming Cheung Kevin Tong
Communications Systems Research
238-420
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109

Abstract. The Galileo spacecraft is currently on its way to Jupiter and its moons. In April 1991, the high gain antenna (HGA) failed to deploy as commanded. In case the current efforts to deploy the HGA fails, communications during the Jupiter encounters will be through one of two low gain antenna (LGA) on an S-band (2.3 Ghz) carrier. A lot of effort has been and will be conducted to attempt to open the HGA. Also various options for improving Galileo's telemetry downlink performance are being evaluated in the event that the HGA will not open at Jupiter arrival. Among all viable options the most promising and powerful one is to perform image and non-image data compression in software onboard the spacecraft. This involves in-flight re-programming of the existing flight software of Galileo's Command and Data Subsystem processors and Attitude and Articulation Control System (AACS) processor, which have very limited computational and memory resources. In this article we describe the proposed data compression algorithms and give their respective compression performance.

The planned image compression algorithm is a 4×4 or an 8×8 multiplication-free integer cosine transform (ICT) scheme, which can be viewed as an integer approximation of the popular discrete cosine transform (DCT) scheme. The implementation complexity of the ICT schemes is much lower than the DCT-based schemes, yet the performances of the two algorithms are indistinguishable.

The proposed non-image compression algorithm is a Lempel-Ziv-Welch (LZW) variant, which is a lossless universal compression algorithm based on a dynamic dictionary lookup table. We developed a simple and efficient hashing function to perform the string search.

1. Introduction

The Galileo spacecraft, which was launched in Oct 1989, is now on its way to Jupiter. Its mission includes releasing a probe into the Jovian atmosphere, Io flyby, probe data capture and relay, Jupiter orbital insertion, and 10 satellite encounters (Ganymede, Callisto, Europa). The Galileo project involves over 20 years of effort. In April 1991, when the spacecraft first flew by Earth, the Galileo team commanded the spacecraft to open the 1.8m high-gain antenna (HGA). However, the HGA failed to completely deploy. All indications are that 3 of the 18 ribs are stuck to the antenna's central tower. Several unsuccessful attempts have been made to free the stuck ribs. A major effort is planned for December 1992 to perform hammering or pulsing of the deployment motor to try to free the ribs. If the HGA fails to deploy, the only way to communicate between

* The research described in this paper was carried out by Jet Propulsion Laboratory, California Institute of Technology, under a contract with National Aeronautics and Space Administration

Earth and the spacecraft is through the use of one of the two low gain antennas. If the current configuration (ground and spacecraft) remains unchanged, the telemetry data rate will be 10 bits per second at Jupiter arrival (1995), compared to the expected data rate of 134 kbits per second in the HGA configuration. The amount of data that can be returned would be drastically reduced.

A study [1] was conducted from December 1991 through March 1992 to evaluate various options for improving Galileo's telemetry downlink performance in the event that the HGA does not open by Jupiter arrival. Among all viable options the most promising and powerful one is to perform image and non-image data compression in software onboard the spacecraft. This involves in-flight re-programming of the existing flight software of Galileo's Command and Data Subsystem (CDS) processors and the Attitude and Articulation Control System (AACS) processor, which has severely limited computational and memory resources. The software has to be compact and computationally simple. A lossy image compression scheme is proposed that can give a wide range of rate-distortion trade-off for the image data, represents over 70% of the data to be returned by the mission. The rest of the data comes from various spacecraft instruments. This can either be compressed by using instrument-specific compression schemes or by using a proposed lossless universal compression algorithm. In this article we describe the proposed image compression scheme and the universal lossless compression algorithm and give their respective compression performances.

The proposed image compression algorithm is a 4×4 or an 8×8 multiplication-free integer cosine transform (ICT) [2], which was first proposed by Cham. The ICT can be viewed as an integer approximation of the popular discrete cosine transform (DCT) scheme. The 8×8 multiplication-free ICT will be implemented in software using the more powerful AACS processor and the 4×4 ICT will be implemented in software using several CDS processors as backup. The ICT schemes have much lower implementation complexity and give indistinguishable performances when compared to the DCT schemes.

The proposed non-image compression algorithm is a Lempel-Ziv-Welch (LZW) variant [3], which is a lossless universal compression scheme. Due to the severe limitations of the CDS processors, we cannot use the more sophisticated existing hashing functions [3]. We developed a simple and efficient hashing algorithm to perform string search. This hashing function uses a total 1802 bytes of memory for a codebook of size 512 bytes, and requires on the average only 3 16-bit comparisons per input byte.

The communication scenario described in this article is unique. Rather than a typical data compression paradigm as in industry where a sophisticated encoder and simple decoder are desired, the Galileo HGA anomaly situation requires a very simple compressor onboard the spacecraft. The decompressor, which is on the ground, can be reasonably complex. Many of the compression techniques described in this article are not novel and are modifications and enhancements of some existing algorithms to adapt to the HGA anomaly scenario. The main goal is to simplify the onboard compressor implementation.

The rest of this article is organized as follows: Sections I, II, III, IV, V, VI and VIII describe the ICT lossy image compression scheme. A more in-depth discussion of the relationship between ICT and DCT is given in Section II. The interplay between the orthonormal transform stage (any orthonormal transform, not just DCT) and the quantization stage is explored in Section III. The mathematical properties and a general construction scheme for the multiplication-free ICT matrix are given in Section IV. A general construction procedure of ICT matrix is described in SECTION

V. Examples of 4×4, 8×8, and 16×16 ICT matrices are given in Section VI. The rate-distortion performance of the ICT schemes for the Galileo S-Band Contingency Mission (4×4 and 8×8) is described in Section VII. Section VIII gives an overview of the LZW algorithm. Section IX describes the LZW scheme we used in the Galileo LGA mission. Section X describes the novel features of the Galileo LZW implementation.

2. DCT Versus ICT

The discrete cosine transform (DCT) is regarded as one of the best transform techniques in image coding. Its independence from the source data and the availability of fast transform algorithms make the DCT an attractive candidate for many practical image processing applications. In fact the ISO/CCITT standards of image processing in both still-image and video transmissions includes the two-dimensional DCT as a standard processing component in many applications. For still-image compression, the transform-based scheme consists of three stages: the data transform stage, the quantization stage and the entropy-coding stage. For video compression, an additional motion-compensation stage with feed-back is included. The enormous popularity of the DCT in image compression provides the driving force for researchers to develop efficient hardware and software implementations for the DCT.

The commercial acceptance of the emerging JPEG and MPGE Standards, which uses an 8x8 block DCT has created a need for an efficient DCT algorithm. A lot of effort has been devoted to the pursuit of reducing the computational complexity of the DCT. New algorithms have already been proposed[6][7]. The idea of incorporating the scale-factors of the transform process as part of the scalar quantizer has also been suggested in the recent literature [6][7]. All these efforts gear towards reducing the total number of floating-point or fixed-point multiplications and additions used, with the emphasis on reducing the number of multiplications.

Recently Cham [2] took a different approach and proposed a new 8-point transform called the integer cosine transform (ICT). ICT requires only integer multiplications and additions, making it much simpler to implement than the DCT. An ICT chip was fabricated and was proven to be efficient in both silicon area and speed. The elements in an ICT matrix are all integers, with sign and magnitude patterns that resemble those of the DCT matrix. The similarity of the ICT matrix to the DCT matrix, together with the orthogonality property of the ICT ($CC^t = \Delta$, where C is an ICT matrix and Δ is a diagonal matrix), guarantee that the ICT as well as its inverse possess the same transform structure as the DCT, thus allowing the use of some fast DCT algorithms to compute a fast ICT [2].

Although the 8-point ICT proposed by Cham performs remarkably well, it is quite ad hoc. In this article we put ICT into a more formal mathematical setting and generalize Cham's idea to any N-point ICT. The mathematical properties of orthonormal transforms including ICT are investigated in the following sections. Since ICT is separable and the extension of the one dimensional ICT to two dimensions is straight-forward, this article focuses on the one dimensional case.

3. Orthonormal Versus Orthogonal Separable Transforms

An $N \times N$ 1-D matrix M is said to be orthonormal if and only if $MM^T = I$, where I is the identity matrix. An $N \times N$ 1-D matrix C is said to be orthogonal if $CC^T = \Delta$, where Δ is a diagonal matrix. It can be shown from basic linear algebra that for any $N \times N$ orthogonal matrix C, there exists an $N \times N$ orthonormal matrix M and an $N \times N$ diagonal matrix Δ such that $M = \sqrt{\Delta}C$. It can further be shown that $C^{-1} = C^T\Delta$ and $M^{-1} = C^{-1}\sqrt{\Delta}^{-1} = C^T\Delta\sqrt{\Delta} = C^T\sqrt{\Delta}$.

The corresponding 2-D $N^2 \times N^2$ orthonormal separable transform matrix is

$$M \otimes M = (\sqrt{\Delta}C \otimes \sqrt{\Delta}C) = (\sqrt{\Delta} \otimes \sqrt{\Delta})(C \otimes C), \tag{1}$$

where $X \otimes Y$ denotes the tensor product of the matrix X with Y, and the corresponding 2-D $N^2 \times N^2$ orthonormal inverse transform matrix is

$$(M \otimes M)^{-1} = (M^{-1} \otimes M^{-1}) = (C^T\sqrt{\Delta} \otimes C^T\sqrt{\Delta}) = (C^T \otimes C^T)(\sqrt{\Delta} \otimes \sqrt{\Delta}), \tag{2}$$

The matrix $\sqrt{\Delta} \otimes \sqrt{\Delta}$ is diagonal. Therefore when the 2-D orthonormal transform $M \otimes M$ is followed by quantization, the diagonal matrix $\sqrt{\Delta} \otimes \sqrt{\Delta}$ can be absorbed in the quantization stage and, only the product by the orthogonal matrix $C \otimes C$ is computed in the transform stage. Similarly on the decoder side, $\sqrt{\Delta} \otimes \sqrt{\Delta}$ can be absorbed in the de-quantization stage, and the N^2 output samples from the de-quantizer are multiplied by the orthogonal matrix $C^T \otimes C^T$. The fusion of the scaling factors of the transform (inverse) transform stage into the quantization (de-quantization) stage does not require additional computation, since division operations have to be performed in the quantization process anyway. An example of a quantization stepsize template that corresponds to the all-one uniform quantization template for an 8×8 ICT is given in Figure 2. A more detailed discussion on incorporating the scale-factors of the transform process as part of the scalar quantizer can be found in [7]. This relaxation of the orthonormal requirement to orthogonal requirement play a crucial role in allowing one to "integerize" a transform coding scheme as we will see in the next section.

4. Mathematical Properties of ICT

ICT and DCT are closely related. Let C and A be the respective ICT and DCT $N \times N$ matrices. $A = [a_{kn}]$ is an orthonormal matrix (i.e. $AA^t = I$) defined as follows:

$$\begin{aligned} a_{kn} &= \frac{1}{\sqrt{N}} \qquad k = 0,\ 0 \le n \le N-1 \\ &= \sqrt{\frac{2}{N}} \cos\frac{\pi(2n+1)k}{2N} \qquad 1 \le k \le N-1,\ 0 \le n \le N-1 \end{aligned} \tag{3}$$

Using A as a template, the ICT matrix $C = [c_{kn}]$ is an orthogonal matrix (i.e. $CC^t = \Delta$, where Δ is a diagonal matrix) with the following properties:

1. Integer property: c_{kn} are integers for $0 \le k, n \le N-1$.

2. Orthogonality property: Rows (or columns) of C are orthogonal.

3. Relationship with DCT: (i) $sgn(c_{kn}) = sgn(a_{kn})$ for $0 \leq k, n \leq N-1$. (ii)If $a_{kn} = a_{st}$, then $c_{kn} = c_{st}$ for $0 \leq k, n, s, t \leq N-1$.

The integer property eliminates real multiplication and real addition operations. The orthogonality property assures that the inverse ICT has the same transform structure as the ICT. Notice that C is only required to be orthogonal, but not orthonormal. However, any orthogonal matrix can be made orthonormal by multiplying it by an appropriate diagonal matrix. This operation can be incorporated in the quantization (dequantization) stage of the compression (decompression) scheme, thus sparing the ICT (inverse ICT) transform from floating-point operations, and at the same time preserving the same transform structure as in the floating-point DCT (inverse DCT). The relationship between ICT and DCT guarantees efficient energy packing and allows the use of some fast DCT technique for the ICT.

5. A General Procedure to Construct an ICT Matrix

A general procedure to construct an $N \times N$ ICT matrix is presented in this section. For any $N \times N$ ICT matrix, this construction is done on the ground prior to implementation. The DCT matrix is used as a template to generate an ICT matrix. The procedure is described as follows:

1. Generate the $N \times N$ DCT matrix A.

2. Construct an $N \times N$ matrix C by substituting the N possible absolute values in A with N symbols, and preserve the signs of the elements in A.

3. Evaluate CC^t, and generate a set of independent algebraic equations which forces CC^t to be a diagonal matrix.

4. Find a set of N numbers which satisfies the set of algebraic equations generated in part 3.

Since for a given N, there are $N(N-1)$ non-diagonal elements in C, part (3) gives $N(N-1)/2$ quadratic equations. This set of equations is too large to be handled easily except for small N. The most tedious part of the above procedure is part 4, that is finding N integers satisfying the set of non-linear algebraic equations generated in part 3. By using advanced symbolic manipulation tools like *Mathematica* [8], the effort to generate the set of algebraic equations in part 3 and solving them in part 4 can be greatly reduced. In fact *Mathematica* was used in an interactive manner to generate a 4×4, an 8×8 and a 16×16 ICT matrices as described in the next section.

In order to obtain good compression performance one requires the set of $N-1$ integers to have a similar magnitude profile to the $N-1$ floating-point elements of A. Furthermore, if the multiplication-free property is desired, one has to restrict the set of N integers to be small integers, so that any multiplications with the matrix elements can be replaced by a small number of adds and shifts.

6. Examples of ICT Matrix Construction

Using the construction techniques described in the previous section, we generated a 4×4 (Figure 1), an 8×8 (Figure 2), and a 16×16 (Figure 3) ICT matrices. The 4×4 ICT matrix has

elements which are powers of 2. The 8×8 ICT matrix is the same as the example given in Cham's paper [2], whose elements are either powers of 2, or are sums of two powers of 2.

7. Compression Performance of the ICT Schemes

We applied our implementation of the 4×4 and the 8×8 ICT schemes for the Galileo S-Band Contingency Mission. We compressed a typical planetary image *miranda* (moon of Uranus). For the purpose of comparison, we also compressed the same image using the JPEG schemes. The root-mean-square-error (RMSE) versus compression ratio performances of these schemes on *miranda* are given in Figure 4. These simulation results indicate that the difference in rate-distortion performance resulting from using the floating-point DCT or the ICT is unnoticeable.

The ICT schemes are also being considered for compression of non-image data like the multi-spectral plasma wave spectrometer (PWS) data. We compressed some typical PWS data files by a factor of 10, which results in lossy reconstructed images that can still be useful for PWS analysis.

8. LZW Overview

The universal lossless LZW algorithm used in this mission is based on the algorithm proposed by Terry A. Welch[3].

The LZW algorithm is an adaptive compression scheme which converts a variable length string into a fixed length string. The algorithm is adaptive in the sense that it uses a dynamic lookup dictionary table. The table is dynamic because it initially starts with an empty table of symbol strings and the algorithm fills this table during the compression and decompression process. The table is thus adapted to the incoming data. Because of this adaptation, the algorithm requires no prior information on the data characteristics of the incoming data.

The LZW implementation of the compression and decompression scheme is based on Welch's paper[3] with modifications to handle multiple dictionary tables, a more efficient search algorithm and the ability to detect certain errors. However, it must be noted that there is an error in the decompression algorithm described in Welch's paper. If followed exactly, the decompressed data will be garbled at random points. This error is located in the "special case" part of decompression algorithm defined in Welch's paper. Instead of a direct output of the decoded final character, this character should be pushed onto the stack.

Our contribution in this paper is to develop an LZW scheme that is efficient in terms of speed and compression performance and at the same time satisfies the stringent computation and memory constraints of the spacecraft.

9. LZW Algorithm

The LZW algorithm is organized around a translation table, referred to as a dictionary table that maps strings of input symbols into fixed length codes. In this particular mission, the code size used is 9-bits, which translates into a table size of 512. The dictionary is used as a lookup table in

both the compression and decompression and is generated as the data is being processed. If the required information is not in the present state of the table, a new entity is added to the table, thus a dynamic lookup table.

The compression speed is very sensitive to the search of the dictionary table in the main loop of the LZW algorithm. The search is used to determine if the required information is in the table. Since the entire LZW algorithm is based on the state of this table, it is important to develop a fast search routine that is also efficient in memory usage because of the memory constraint of the spacecraft.

9.1 The Dynamic Lookup Dictionary Table

The size of the dictionary has a direct bearing on the memory requirements and execution time of the implemented program. The proposed dictionary size for this mission is 512. This number is a compromise between optimal compression and the memory constraint on board the spacecraft. The increase in dictionary size from 512 to 1024 does not produce a great enough compression gain to justify choosing the larger dictionary size.

10. Features of the Galileo LZW Implementation

The LZW algorithm was implemented with features that were not discussed in Welch's paper. The implementation can concurrently compress multiple independent streams of data using multiple dictionaries while using the a minimal amount of memory without compromising execution time.

10.1 Multiple Dictionary Tables

The multiple dictionary table feature was added because the spacecraft transmits different types of data requiring lossless compression. Examples of these types of data are telemetry, engineering and instrument data. Using the multiple dictionary approach, it is possible to segregate these data streams without requiring the compression algorithm to finish up on one stream and start another table. The program can switch back and forth between the data streams and use the dictionary table that is assigned to that data stream.

10.2 The Hashing Algorithm

The search portion of the LZW algorithm is the most time consuming, thus it was necessary to design a search procedure that was both efficient in memory and in execution time. We employ a simple yet efficient hashing algorithm to perform the search. Normal implementation of hashing uses dynamic memory and a linked list, but in our implementation, two fixed arrays are used. This is to save memory and to save overhead time in keeping track of the linked list using dynamic memory. The size of the first array is equal to the dictionary size and the second array would equal the difference between the dictionary size and the alphabet size. Thus it would require one array of size 256 and the second array of size 512 for a dictionary size of 512. See Table 1 for hashing performance.

10.3 LZW Performance on Near-Infrared Mapping Spectrometer (NIMS) Data

We have obtained NIMS data produced by Galileo to test the performance of the LZW implementation. See Table 2.

10.4 LZW Performance on Selected Text Data

A text file was produced and used to show the performance of the LZW algorithm with various table sizes. (See Table 3.) From the performance, we can see that the table size proposed is a good compromise between optimal compression and memory usage.

Acknowledgment

The authors would like to thank F. Pollara (Section 331) for his input to this work, and S. Dolinar (Section 331) and L. Swanson (Section 331) for their constructive comments. The authors would also like to thank T. Brady (Section 348) for his assessment on the implementation feasibility of the compression schemes described in this article. Particular thanks to the Deep Space Network Advance System Program whose support makes this project possible.

References

[1] L. Deutsch and J. Marr, "Galileo S-Band Mission Study, Final Report," JPL Publication, Mar 1992.

[2] W. Cham, "Development of Integer Cosine Transforms by the Principle of Dyadic Symmetry," *IEE Proceedings*, Vol. 136, Pt. I, No. 4, August 1989.

[3] T. Welch, "A Technique for High performance Data Compression," *Computer*, June 1984.

[4] Joint Photographic Experts Group (JPEG) Draft Technical Specification (Rev. 5), ISO/CCITT, January 15, 1990.

[5] Moving Picture Experts Group (MPEG) Draft Technical Specification, ISO/CCITT, May, 1991.

[6] Y. Arai, T. Agui, M. Nakajima, "A Fast DCT-SQ Scheme for Images," *Trans. of the IEICE*, vol.E 71, pp.1095-1097, November, 1988.

[7] E. Feig and S. Winograd, "Fast Algorithms for the Discrete Cosine Transform," submitted to the IEEE Transactions on Acoustics, Speech, and Signal Processing.

[8] S. Wolfram, *Mathematica : A System for Doing Mathematica by Computer*, Addison-Wesley Publishing Company, 1988.

1	1	1	1
2	1	-1	-2
1	-1	-1	1
1	-2	2	-1

Figure 1 a 4 x 4 ICT Matrix

1	1	1	1	1	1	1	1
5	3	2	1	-1	-2	-3	-5
3	1	-1	-3	-3	-1	1	3
3	-1	-5	-2	2	5	1	-3
1	-1	-1	1	1	-1	-1	1
2	-5	1	3	-3	-1	5	-2
1	-3	3	-1	-1	3	-3	1
1	-2	3	-5	5	-3	2	-1

Figure 2a an 8 x 8 ICT Matrix

8	25	18	25	8	25	18	25
25	78	56	78	25	78	56	78
18	56	40	56	18	56	40	56
25	78	56	78	25	78	56	78
8	25	18	25	8	25	18	25
25	78	56	78	25	78	56	78
18	56	40	56	18	56	40	56
25	78	56	78	25	78	56	78

Figure 2b the Quantization Template of 2a

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
42	38	37	32	22	19	10	4	-4	-10	-19	-22	-32	-37	-38	-42
10	9	6	2	-2	-6	-9	-10	-10	-9	-6	-2	2	6	9	10
38	22	4	-19	-37	-42	-32	-10	10	32	42	37	19	-4	-22	-38
2	5	-5	-2	-2	-5	5	2	2	5	-5	-2	-2	-5	5	2
37	4	-32	-38	-10	22	42	19	-19	-42	-22	10	38	32	-4	-37
9	-2	-10	-6	6	10	2	-9	-9	2	10	6	-6	-10	-2	9
32	-19	-38	4	42	10	-37	-22	22	37	-10	-42	-4	38	19	-32
1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1
22	-37	-10	42	-4	-38	19	32	-32	-19	38	4	-42	10	37	-22
6	-10	2	9	-9	-2	10	-6	-6	10	-2	-9	9	2	-10	6
19	-42	22	10	-38	32	4	-37	37	-4	-32	38	-10	-22	42	-19
5	-2	2	-5	-5	2	-2	5	5	-2	2	-5	-5	2	-2	5
10	-32	42	-37	19	4	-22	38	-38	22	-4	-19	37	-42	32	-10
2	-6	9	-10	10	-9	6	-2	-2	6	-9	10	-10	9	-6	2
4	-10	19	-22	32	-37	38	-42	42	-38	37	-32	22	-19	10	-4

Figure 3 a 16 x 16 ICT Matrix

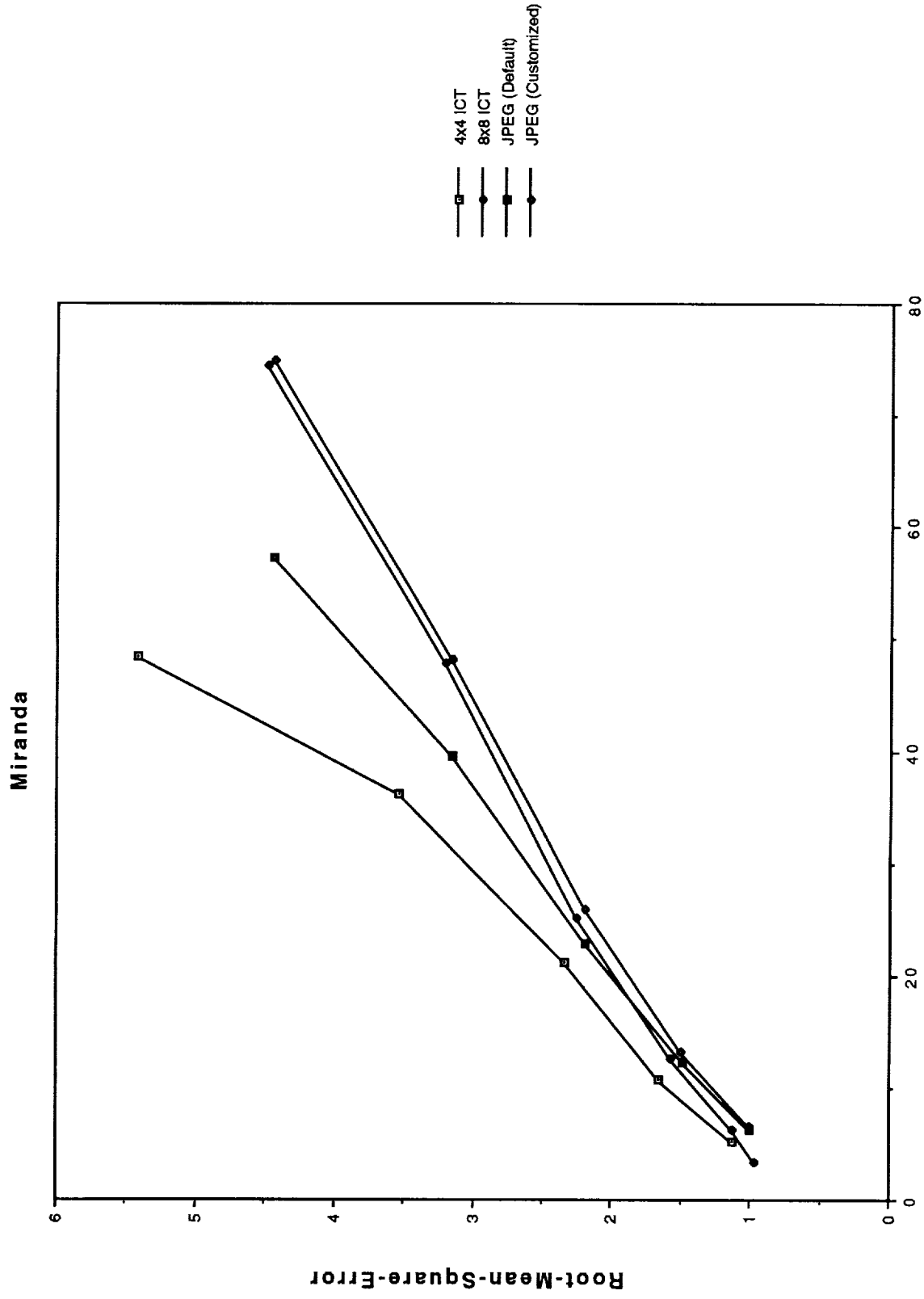

Figure 4 Compression Performances

TABLE 1 Hashing Comparison

Data files used are 256x256 = 65536 bytes planetary image files.

Compares Per Input Byte (Table 1)

	Hash Algorithm Search TableSize		Sequential Search TableSize	
File	512	1024	512	1024
d1	3.2436	3.7604	76.0222	184.5167
f2	2.5246	3.0067	73.0339	178.2090
h2	2.5816	3.1000	74.2097	181.3555
l2	3.6721	4.8428	93.6538	233.7537

TABLE 2 Nims Data Performance

COMPRESSION RATIO (Table 2)

Data Orientation	Table Size = 512	Table Size = 1024
Horizontal Scan	2.60	2.69
Vertical Scan	2.59	2.63
Mirror Scan	2.27	2.35
Original Data	2.45	2.51

TABLE 3 Text Data Performance

Sample Text File Size = 5390 bytes (Table 3)

Table Size	Compression Ratio	Tables Used
512	1.36	14
1024	1.52	4
2048	1.59	2