IMAGE COMPRESSION SOFTWARE FOR THE SOHO LASCO AND EIT EXPERIMENTS

Mitchell R Grunes
ALLIED-SIGNAL TECHNICAL SERVICES
Russell A Howard
Karl Hoppel
Stephen A Mango
NAVAL RESEARCH LABORATORY
Dennis Wang
INTERFEROMETRICS

ABSTRACT

This paper describes the lossless and lossy image compression algorithms to be used on board the Solar Heliospheric Observatory in conjunction with the Large Angle Spectrometric Coronograph and Extreme Ultraviolet Imaging Telescope experiments. It also shows preliminary results obtained using similar prior imagery and discusses the lossy compression artifacts which will result. This paper is in part intended for the use of SOHO investigators who need to understand the results of SOHO compression in order to better allocate the transmission bits which they have been allocated.

INTRODUCTION

The Solar and Heliospheric Observatory (SOHO) is currently scheduled for a July 1995 launch into a lunar L1 orbit. The software described will compress images from the Large Angle Spectrometric Coronograph (LASCO) (a wide-field white light and spectrometric coronograph) and the Extreme Ultraviolet Imaging Telescope (EIT) experiments. LASCO will image the solar corona from about 1.1 to 30 solar radii, and has a built in spectrometer to measure, point-by-point, plasma temperature, density, bulk and turbulent velocities, and the direction of the magnetic field.

The transmission bandwidth (5200 bits/sec) is insufficient to transmit the desired imagery. In order to resolve this problem, our software implements two image compression algorithms:

1. A lossless image compression algorithm.
2. A lossy image compression algorithm, expected to be used for most of the imagery. In most cases investigators are expected to select an output of about 1.6 bits/pixel (bpp), a compression factor of 10 from the input 16 bit format. This will allow transmission of about 240 images/day, plus some other overhead and small transient images.

The code is mostly written in the C programming language. It will run on a Sandia SA3300 CPU, a rather slow (about 1 MIPS) radiation hardened space qualified processor which was designed to emulate a National Semiconductor 32C016 Series CPU.

The relatively slow data rate allows us to use compression algorithms which are of higher quality on the solar test imagery than published standards such as JPEG, in spite of the hardware limitations of the target computer. This was accomplished at the cost of increased complexity and processing load. However, these are acceptable for our application because:

1. The data will be gathered at substantial cost.

2. As in many space applications, the allotted transmission bandwidth is the major limiting factor on the transmitted spatial and radiometric resolution, and on the frequency with which images can be transmitted. This is because transmission bandwidth translates directly to power and storage requirements, and thus to the weight and cost of the satellite.

As in many space applications, the imagery will be reconstructed (decompressed) by a work station on the ground with much more computing power than the compressing computer.

Some comparisons with the independent JPEG algorithm will also be given.

This statistics that appear in this paper are somewhat preliminary. The final paper may use somewhat different algorithms which may produce better results. In particular, several changes to our algorithms will be investigated in order to insure that the result is as close to the optimal as is practical within the constraints of the target processor. For lossless compression this might include the use of a non-integral number of bits to code the least significant fraction of the split coder, or the use of adjusted binary codes after the style of Golomb. It is not clear at this time what this might include for lossy coding.

LOSSLESS COMPRESSION ALGORITHM

The method described in

> Rice, "Some Practical Noiseless Coding Techniques, Part III, Module PSI14,K+", JPL Publication 91-3, 11/91

served as a starting point for the development of the lossless compression algorithm because:

1. It requires relatively little code or time to implement.
2. Very few bits are needed to provide small block size adaptivity. This is important because there is expected to be a great deal of difference in brightness and texture between different parts of the image, and because CCD array sensors develop small area defects.

Various changes were made to that algorithm. In brief:

1. Different choice of block size, and the use of bi-level two dimensional blocks.
2. More adaptive classes.
3. Triplet coding was not implemented because it is anticipated that the 14 to 16 bit images will be statistically random in the lower few bits.
4. A somewhat improved prediction algorithm.
5. A somewhat more complex coding technique was used to keep down the number of bits used for adaptivity.

An optimal DPCM technique was also investigated. The weights were determined by a least squares fit. This produced predictions which were then input into the modified Rice algorithm. This improved the compression factor by only 5% for the 13 bit eclipse image. The improvement will probably be even smaller for the 14 and 16 bit imagery that the software will be applied to. Hence it was decided that it was not worth performing least squares processing to determine optimal weights.

Small scale adaptivity outweighs the advantages of more sophisticated entropy coding. For example, it significantly out-performs pure Huffman coding techniques on sample images similar to those expected from SOHO. In fact it performs somewhat better than would appear to be

possible on the basis of *whole-image* "entropy" measured in terms of the frequency of original pixel values, or in terms of the differences from predicted pixels. Note, however, that some methods, such as lossless JPEG, do produce better results for many 8 bit images. It is quite possible that a better algorithm may be used in the final software.

LOSSY COMPRESSION ALGORITHM

The ADCT (Adaptive Discrete Cosine Transform) method described in

> Chen and Smith, "Adaptive Coding of Monochrome and Color Images", IEEE vol 25 #11, Nov 1977, pp. 1285-1292

served as a starting point for the development of the lossy compression algorithm because:

1. It is a method with which NRL Code 7230 has a great deal of experience. We have implemented that algorithm (somewhat differently) in a software package which has been used operationally for some time by various U.S. government agencies.
2. It is a fully adaptive ADCT, which chooses the number of bits used to specify each DCT transform coefficient within each class of block according to its activity. No a priori statistics are required.
3. Max-Lloyd Gaussian quantization is used in the frequency domain, which performs much better than uniform quantizers.
4. One may specify a definite compression factor can be specified over a large, fairly continuous range.
5. It is not especially fast or simple, but it is certainly faster than known high quality fractal and vector quantization algorithms.
6. It remains one of the very best image compression methods yet developed, performing better than many of the more recently published algorithms.

Various changes were made to that algorithm, some of which improve upon our earlier work. In brief:

1. A different block size was chosen, to improve quality, and to mesh better with other intended spacecraft processing.
2. More block classes (up to 16, depending on image size) and a somewhat different method of separating classes (a compromise between block variance and maximum coefficient scaling) is used. These changes were done in order to largely eliminate the discontinuities in brightness and texture that occurred across block boundaries, at the price of somewhat larger RMS pixel errors.
3. The quantization tables are normalized somewhat differently.
4. Very low intensity coefficients are randomized to prevent systematic quantization errors leading to bright or dark spot artifacts.
5. Several details not specified by Chen and Smith were provided by us, such as:
   a. The bit allocation table is sent efficiently, employing run length encoding of alternate direction diagonals.
   b. The coefficients are scaled so as to emphasize the most visible features.

The modified algorithm produces surprisingly good results. In particular, the existence and position of edges remains accurate up to fairly high compression factors (but some blurring occurs, there are echoes and shifts in the radiometric centers of isolated bright points, and there

are some discontinuities at block boundaries). Preliminary work using full search vector quantizations did not yield as good results. Wavelet transform methods might produce more continuous results across block edges, but that did not generally appear to be a problem for the sample images at the desired compression factors.

As a test, the eclipse image was compressed and reconstructed using the lossy algorithm. The difference image was then compressed using lossless compression. The total number of bits used was about the same as to code the image using lossless compression alone. Therefore the lossless and lossy algorithms store about the same amount of information per bit.

APPROPRIATE IMAGERY AND COMPRESSION FACTORS

The software was written to apply to 2 dimensional continuous tone monochrome still imagery, with up to 16 bits/pixel. A number of arbitrary factors in the design were decided on the basis of the solar test imagery.

Both the lossless and lossy compression algorithms perform best with images which are somewhat smooth. For example, they will not perform very well with images that have been digitized in a small number of bits or quantized at a small number of levels, such as dithered images, nor with extremely noisy images, such as one-look SAR.

Both the lossless and lossy compression algorithms perform sub-optimally on images which are so smooth that a significant fraction of pixels are perfectly predictable from their neighbors; the 14-16 bit quantization of our input data will probably contain noise or small scale features in the lower few bits.

The lossy algorithm performs sub-optimally on isolated bright and dark spots or lines, although edges between two regions of differing brightness are represented fairly well. In addition, images containing features with a very wide dynamic range may tend to distort small features with low contrast levels, and some noise is introduced into very low contrast areas. For example, images consisting of many stars or spectral lines would be inappropriate.

If lossless compression is applied to inappropriate images, substantially more bits will be used than are needed. Lossy compression of inappropriate images will blur features, shift the radiometric centers of isolated bright and dark spots, and introduce shape distortions or lose small and subtle features. It may also introduce discontinuities in brightness and texture at block boundaries.

For this project the lossy compression software was intended to be applied at a compression factor of 10 to 15 relative to 16 bit/pixel input, yielding 1.07 to 1.6 output bits/pixel. The algorithm can produce adequate results at somewhat smaller compression factors, and it could theoretically be applied at compression factors up to several hundred. In practice, the inefficiencies due to packet format and small block size make our implementation inappropriate above a compression factor of about 20.

Applying lossy compression with excessive compression factors yields problems similar to applying it to inappropriate images.

DEFINITION OF TERMS

There are a number of terms that we use in evaluating the performance of our software. These terms are defined in many different ways by different researchers.

Compression Factor relative to the 16 bit/pixel input format:

$$CF_{16} = \frac{Bits\ in\ original\ image\ (at\ 16\ bits/pixel)\ with\ no\ overhead}{Bits\ in\ compressed\ image\ with\ overhead\ including\ packets} \quad (1)$$

RMS Error

$$RMS\ Error = \sqrt{Mean\ Square\ (original\ image - reconstructed\ image)}$$

Note that RMS error is very close to standard deviation for both our technique and the independent JPEG algorithm, because systematic bias is negligible in both cases.
Normalized Mean Square Error:

$$NMSE = \frac{Mean\ Square\ (original\ image - reconstructed\ image)}{Mean\ Square\ (original\ image)} \quad (3)$$

Other definitions of NMSE, in which the mean square pixel value is replaced by the maximum or maximum possible value, are quite common. Errors shall be reported both for pixels and for gradients (first differences, taken along both image directions). The former is scientifically meaningful because plasma brightness can be related to total electron content, the latter because feature detection and recognition depends on detection of edges and texture.

Throughout this paper we have omitted the approximately .0625 bits/pixel to be expected in compression packet overhead, as well as the overhead to be used for other types of packets and transmitted information.

TEST IMAGES

We use 5 test images. We shall also test with parts of images masked out. Masking will sometimes be used in the spacecraft to omit parts of the image covered by the occulters. (Occulters are used to eliminate very bright light which would otherwise wash out the desired imagery.) Masking is a very simple form of additional compression, which eliminates the bits needed to code the masked out features.

Image Name	Pixel Columns*Rows*Bits	Actual Source Instrument	Similar to LASCO/EIT telescope
Eclipse	512*512*13	Ground Photograph	C1
Same, masked		" "	C1
Same, masked$_2$		" "	C3
Vidicon	512*512*11	Solar Max	C2
Same, masked		" "	C2
Helio	512*1024*13	HRTS Spectroheliograph	EIT
H_α	1024*1024*14	HRTS H_α	EIT, but lower contrast
Lenna	512*512*8	Human Photograph	None

The HRTS images were summed in 2*2 pixel blocks to reduce the data to the approximate resolution of EIT. Note that the Lenna (sometimes Lena) image has been included simply because it is probably the most commonly used test image in the image processing field. No importance was given to getting good results with Lenna.

All of the test images except Lenna are shown in the figures.

LOSSLESS COMPRESSION RESULTS

Results are first listed for the original test image. 16 bit rescaled values are also very pessimistically estimated by assuming that the additional bits are random. Real imagery should perform better.

Image	CF_{16}	bits/pixel	CF_{16}, rescaled	bits/pixel, rescaled
Eclipse	2.24	7.13	1.58	10.13
Same, masked	2.35	6.80	1.63	9.8
Same, masked$_2$	2.64	6.06	1.77	9.06
Vidicon	3.96	4.04	1.77	9.04
Same, masked	4.64	3.45	1.89	8.45
Helio	1.63	9.80	1.25	12.8
H_α	1.77	9.04	1.45	11.04
Lenna	3.37	4.75	1.25	12.75

LOSSY COMPRESSION RESULTS

Lossy compression, by definition, involves the loss of information. The following table represents the results of compressing the test images to a nominal 1.6 bits/pixel:

Image	Pixel Error				Gradient Error			
	SOHO		JPEG		SOHO		JPEG	
	RMS	NMSE	RMS	NMSE	RMS	NMSE	RMS	NMSE
Eclipse	10.9	1.1E-5	19.2	3.3E-5	13.8	.077	25.6	.263
Same, masked	10.2	9.2E-6	n/a	n/a	13.1	.069	n/a	n/a
Same, masked$_2$	9.3	9.3E-6	n/a	n/a	11.8	.080	n/a	n/a
Vidicon	1.7	8.6E-5	3.7	4.1E-4	2.1	.038	5.3	.241
Same, masked	1.5	5.7E-5	n/a	n/a	1.9	.025	n/a	n/a
Helio	100.5	5.2E-3	114.5	6.8E-3	135.0	.026	173.6	.425
H_α	58.2	1.4E-4	70.9	2.1E-4	85.0	.123	104.1	.185
Lenna	3.5	9.8E-4	3.24	8.3E-4	4.6	.156	5.0	.183

It was not practical to provide JPEG results for the masked images, because the independent JPEG code, as supplied did not implement masks.

Pixel errors are better than those from JPEG, partly because the independent JPEG software was designed to handle 8 bit imagery, so our imagery was scaled to fit. (With real 8 bit test imagery, the results were mixed.) The exception is Lenna, where JPEG does noticeably better. It is our belief that the very extensive use of Lenna, together with RMS error or NMSE, in the compression literature, caused the JPEG and independent JPEG algorithms to be somewhat biased to produce good results with that image.

Gradient errors are uniformly better than those from JPEG, partly for the same reasons, but partly because gradient errors are rarely looked at, so that they probably did not much influence JPEG design.

The figures show that there is very little visual loss. For all images the major apparent change is a blurring of isolated bright and dark points. There is also some noticeable blurring of edges, and there is a modification and introduction of some noise into low contrast features. Overall, however, the compression quality is excellent.

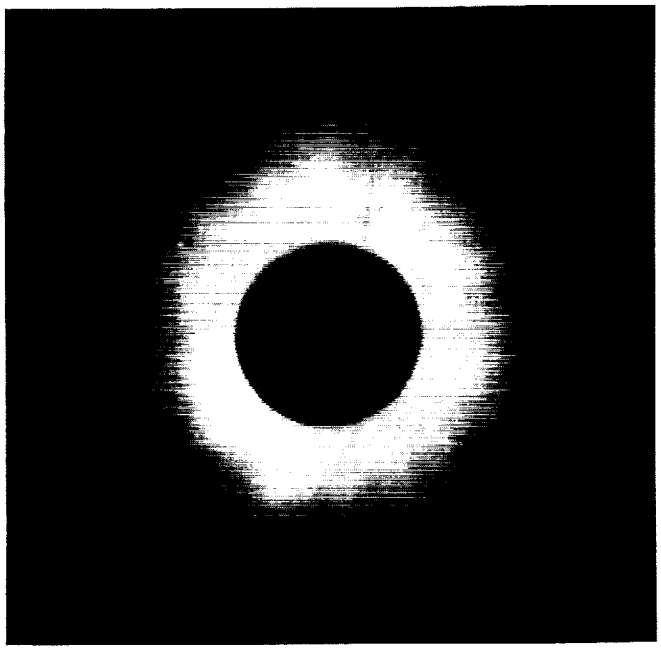

Fig. 1A Original Eclipse image, 512*512 pixels

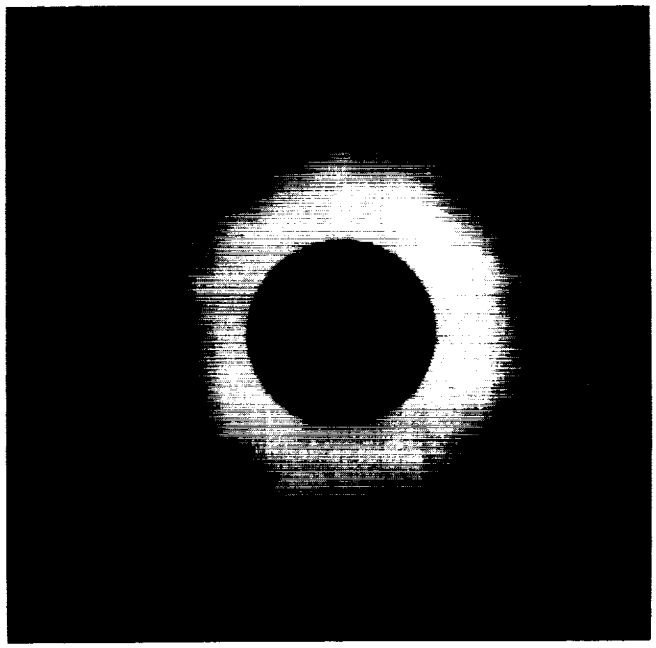

Fig. 1B SOHO compression to about 1.6 bpp

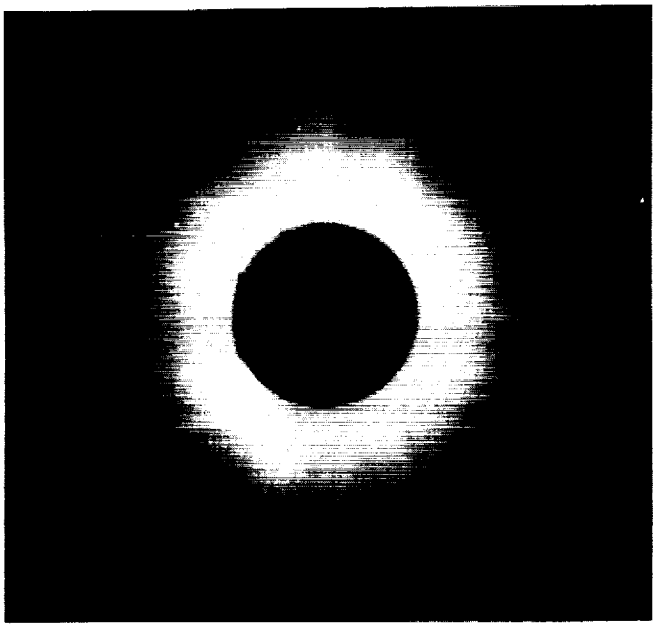

Fig. 1C Same, with C1-like mask

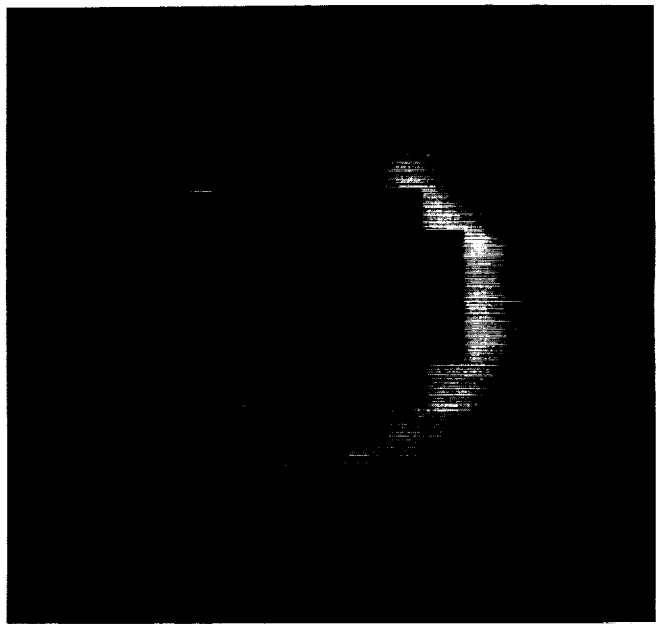

Fig. 1D Same, with C3-like mask

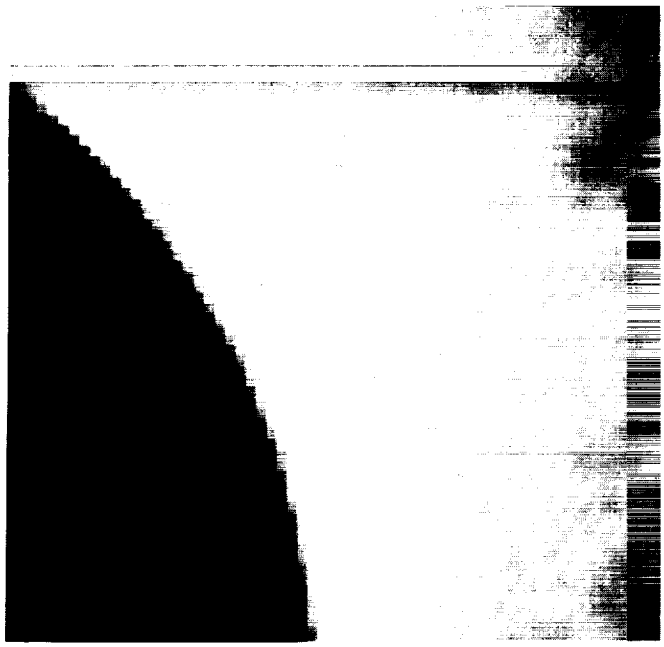

Fig. 2A Stretched 64*64 pixel section of 1A

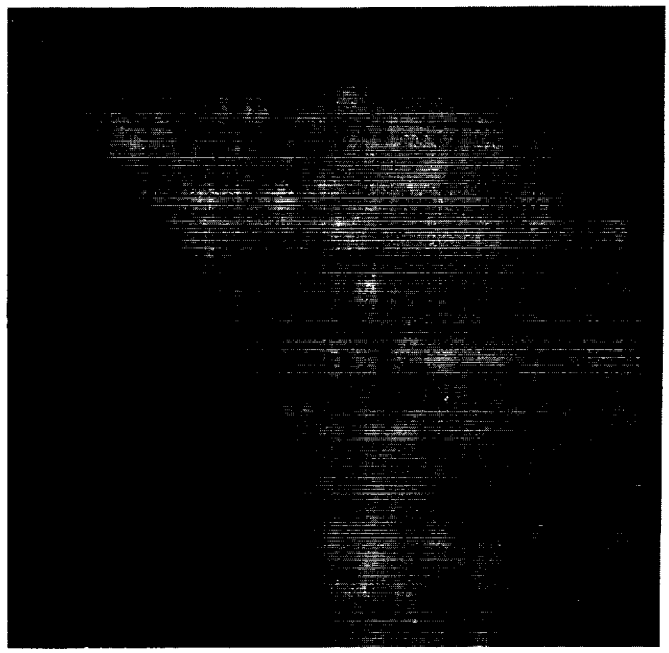

Fig. 2B Same for 1B

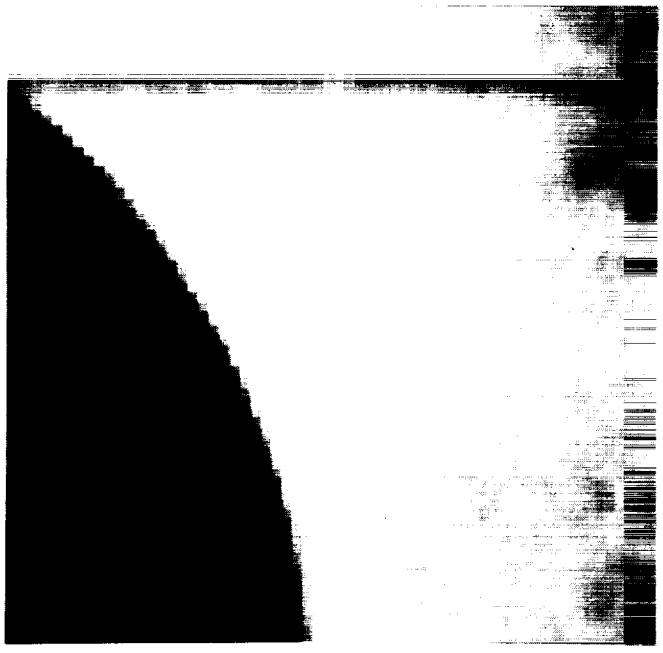

Fig. 2C Same for JPEG compression

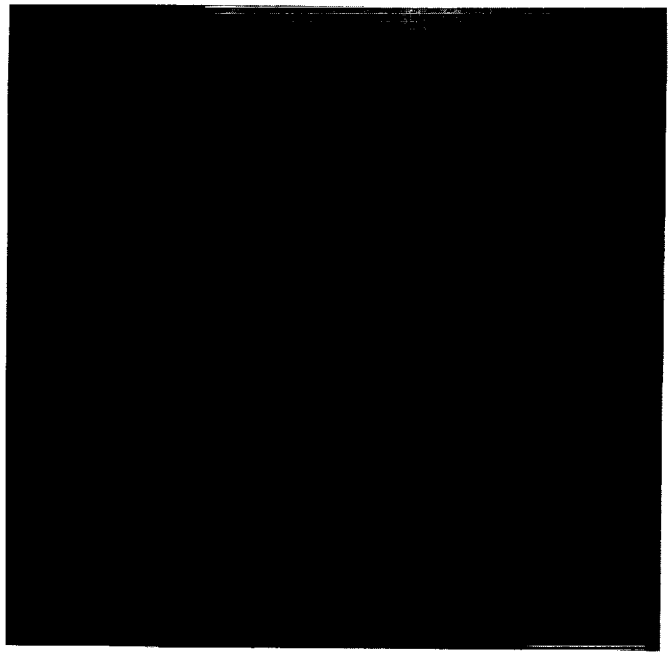

Fig. 2D Same for 1D

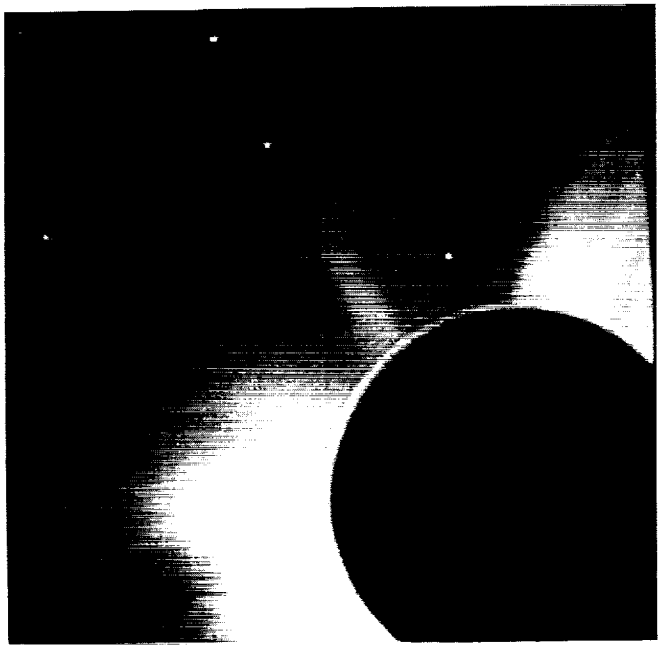

Fig. 3A Original Vidicon image, 512*512 pixels

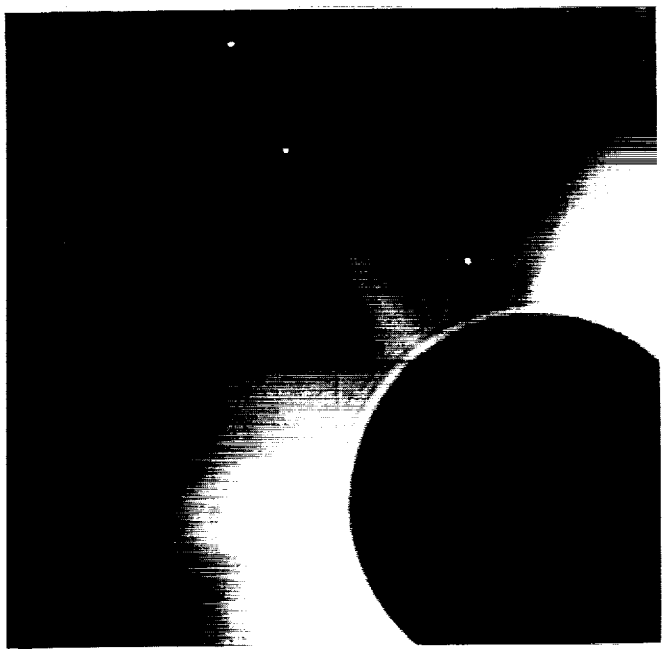

Fig. 3B SOHO compression to about 1.6 bpp

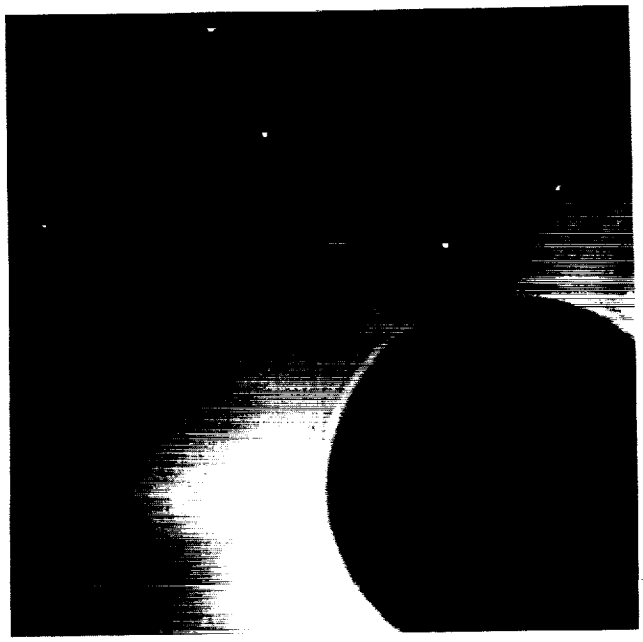

Fig. 3C Same with mask

Fig. 4A Original Helio image, 512*1024 pixels

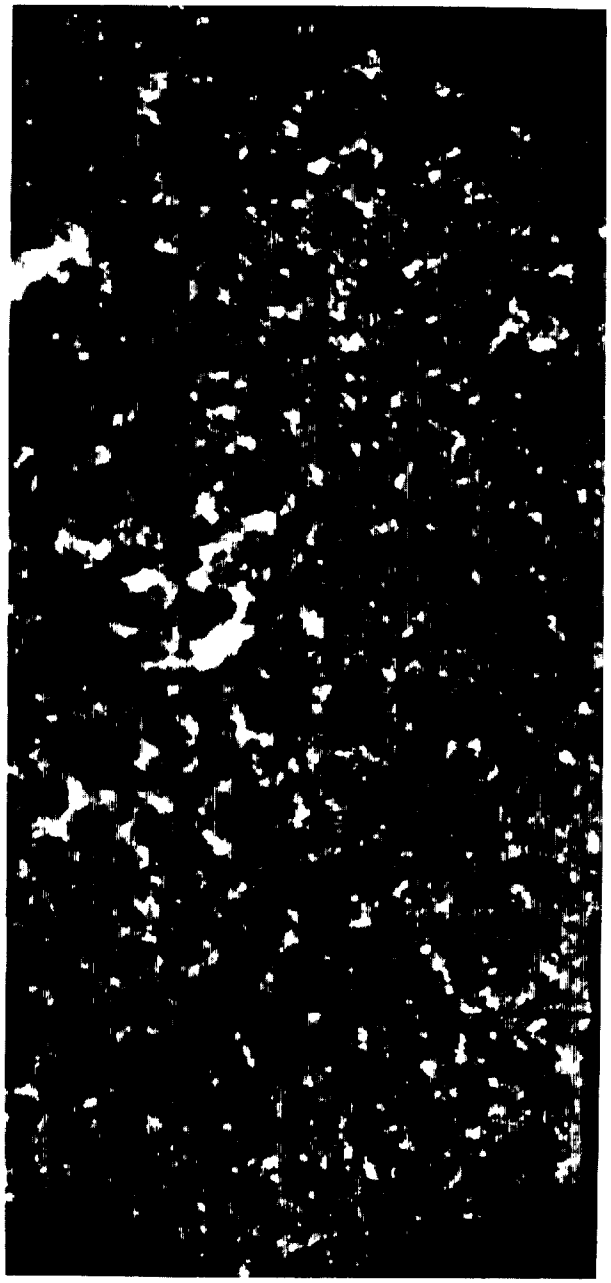

Fig. 4B SOHO compression to about 1.6 bpp

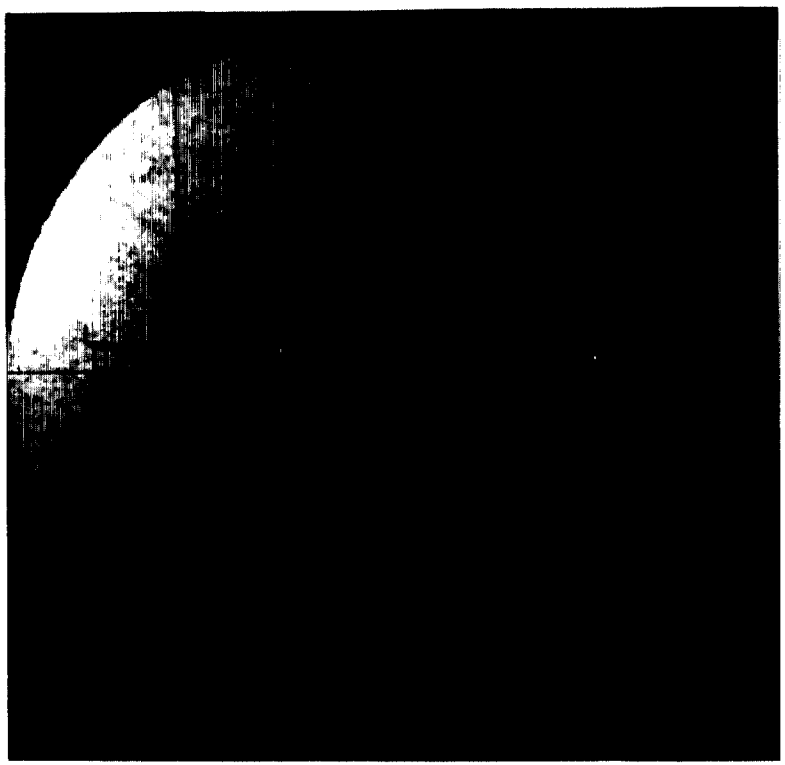

Fig. 5A Original H-alpha image, 1024*1024 pixels

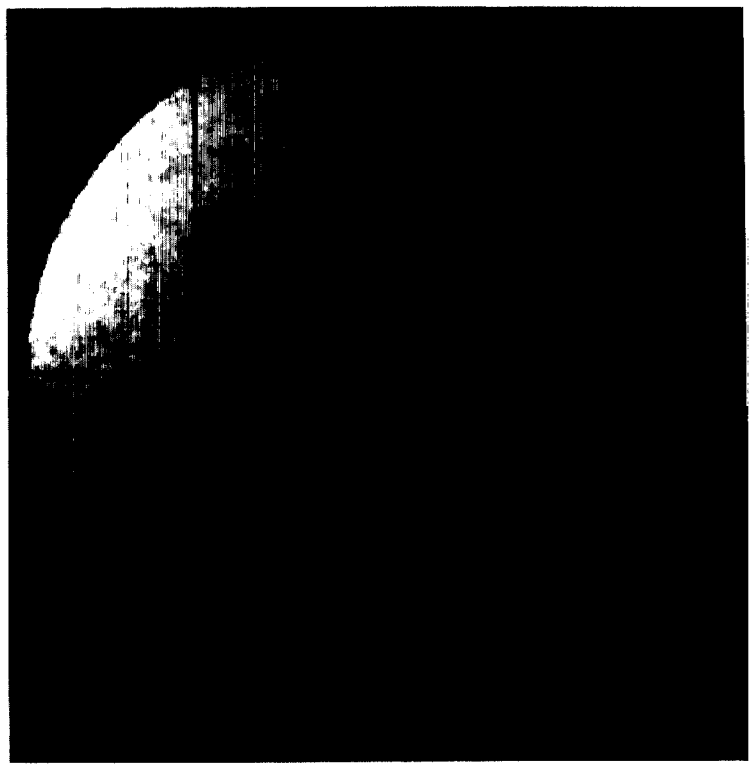

Fig. 5B SOHO compression to about 1.6 bpp