4

GL-TR-90-0128
ENVIRONMENTAL RESEARCH PAPERS, NO. 1063

Recovery of Images From the AMOS ELSI Data for STS-33

DAVID J. KNECHT

19 April 1990

SPACE PHYSICS DIVISION
PROJECT S321
GEOPHYSICS LABORATORY
HANSCOM AFB, MA 01731-5000

REPORT DOCUMENTATION PAGE	Form Approved OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 19 APR 90	3. REPORT TYPE AND DATES COVERED Scientific Interim

4. TITLE AND SUBTITLE Recovery of Images From the AMOS ELSI Data for STS-33	5. FUNDING NUMBERS PE63220C PR S321 TA S32142 WU S3214201
6. AUTHOR(S) David J. Knecht	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Geophysics Laboratory (PHK) Hanscom AFB Massachusetts 01731-5000	8. PERFORMING ORGANIZATION REPORT NUMBER GL-TR-90-0128 ERP, No. 1063
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

During the STS-33 space-shuttle mission in November 1989, ground-based observations of the shuttle were made from the Air Force Maui Optical Station (AMOS) in Hawaii during five orbits that passed near the observatory. The objective of these measurements was the study of interactions between contaminant gases of the shuttle and the residual atmosphere at orbit altitude. The Enhanced Longwave Spectral Imager (ELSI) was one of the instruments used. The ELSI employs two infrared-sensitive array detectors at a telescope focal plane. The recorded digital data from the MWIR detector used in this instance were found to be unusable under ordinary processing methods because five of the twelve bits in each pixel value were found not to be operating. This report describes the further examination of these data, the discovery of other faults in the instrument, and the subsequent development of software routines and a correction algorithm that succeeded in completely recovering the true images contained in the data. Criteria for the success of this method are examined in the course of describing the design of the algorithm. The method has been transfered to the organization that will process similarly flawed classified data from this instrument. Keywords;)

14. SUBJECT TERMS Array Detectors, Image Processing, Correction Algorithms, Data Recovery, Infrared Imagers, Telescope Focal Plane , (KR)	15. NUMBER OF PAGES 42
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

Contents

Illustrations

Recovery of Images from the AMOS ELSI Data for STS-33

1. INTRODUCTION

During the STS-33 space-shuttle mission in late November 1989, a number of instruments of the Air Force Maui Optical Station (AMOS) were used to observe five overflights of the shuttle to study the interaction of spacecraft contaminants with the residual atmosphere. These measurements were part of a secondary shuttle experiment called the AMOS Calibration Tests.

One of the instruments used was the Enhanced Longwave Spectral Imager (ELSI), which is an infrared-sensitive array detector located at a focal plane on the side mount of the 1.6-meter telescope. Data from this instrument are recorded on tape in both video and digital formats. The ELSI was used on three passes, orbits 21, 37, and 67 on 24, 25, and 27 November. These data are normally unclassified, but special circumstances required the ELSI data of 24 and 25 November to be classified. The unclassified data of 27 November, in video format, were handcarried to Geophysics Laboratory (GL) at the beginning of December 1989; the classified data, in digital format, were sent to GL from AMOS some time in February 1990.

Since the Spacecraft Interactions Branch (PHK) does not have facilities for processing classified data, the production of images from the digital tapes was undertaken by the Data Systems Branch (LCY). PHK provided a written description of the tape format, based on information provided by AMOS, and LCY developed the routines needed to read the tapes, but when the data failed to yield recognizable images, the effort was at an impasse. To permit PHK to look into the problem, it was decided to work on the unclassified data of orbit 67, which carries the test number ATN 47434. We requested and received (on 23 March 1990) a one-minute sample of digital data, corresponding to the interval during which the best images were seen on the videotape.

Examination of these data showed that the instrument was suffering from several malfunctions, one of them serious enough to make the data unusable when processed in the usual manner. This report describes the problem and the development of a method to deal with it, specifically, the construction of an algorithm capable of effecting a complete recovery of the images.

(Received for publication 19 April 1990)

2. DESCRIPTION OF THE INSTRUMENT

The ELSI instrument employs two array detectors: a mid-wave (MWIR) detector, which is a gallium-doped silicon array of 45 by 90 pixels, and a long-wave (LWIR) detector, which is a HgCdTe array of 64 by 64 pixels. The image can be moved, by a switch in the control room, from one detector to the other. The spectral range and larger field of view of the LWIR detector make it more useful for acquiring a target; after acquisition the image can be switched to the MWIR detector. The recording for orbit 67 includes data from both detectors, as noted below, but only the MWIR data are of interest here.

The ELSI uses a rapid-chopping scheme to remove the instrument and sky background against which the tracked object is seen. The image is recorded first with the object in the chosen location on the array detector and then again after the object has been deflected slightly by moving one of the flat mirrors in the optical path. Subtracting the second image from the first is extremely effective in producing a clear image of the tracked object against an almost black background. The object is not deflected enough to move it completely off the detector, so the subtraction results in a positive image at the chosen location and a negative image at the deflected position; the latter is removed by setting negative pixel values to zero, leaving only a patch of completely black background. The deflection, which is called *nodding*, is done at a constant nominal rate of 40 cycles per second, implying an image rate of 80 per second.

Analog data from the detector are digitized by equipment mounted on the telescope and then transmitted to the control room, where they are recorded on digital tape. The recording process is further described in the discussion that follows. The digitized data are also sampled and reconverted to analog form, in a standard video format, for display on a video monitor and recording on videotape.

3. TAPE FORMAT AND CONTENT

ELSI digital data are on standard 9-track 6250-bpi tapes. Details of the tape format are shown in Figure 1. Physical records are unformatted binary and contain 16 kilobytes (16,384 bytes), except for the last record in a file, which may be short. A tape may have one or more files, terminated by end-of-file marks, representing different time intervals. Each file begins with a file label of 1024 bytes, followed by an integral number of ELSI logical records of 185,344 bytes each. Therefore, there are more than 11 physical records for each logical record, and boundaries of physical and logical records are not simply related. Each logical record contains, in this order, an 8-byte spacer, twenty images, and a 1016-byte record label.

The present purpose requires no information from the file label or the 8-byte spacers. The only item needed from the record labels is the time, which is given in units of seconds and milliseconds after the reference time of 0000 GMT on 01 January 1970. The seconds component is an unsigned four-byte integer in bytes 21-24; the milliseconds component is an unsigned two-byte integer in bytes 25-26. Pixel values in the image arrays are unsigned two-byte integers. Integers are written to tape starting with the most significant byte.

Each ELSI image contains 9216 bytes or 4608 pixel values in a 6x16x48 *image array*. These are related to the 45x90 MWIR *detector array* as illustrated in Figures 2 and 3. Figure 2 shows how the detector array is laid out in six sections that have separate readouts. Data are recorded on tape in the order in which they come off the detector. The readout takes a value from the same location on each of

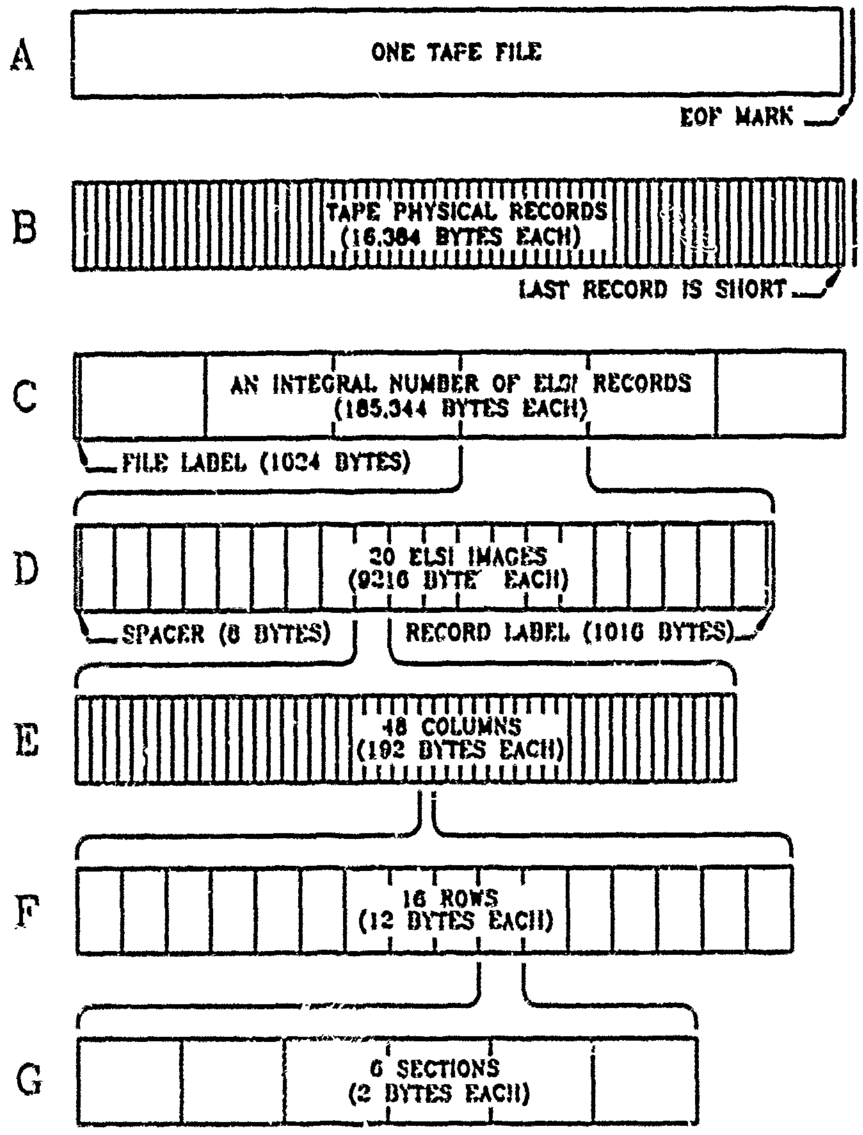

A
ONE TAPE FILE
EOF MARK
B
TAPE PHYSICAL RECORDS (16,384 BYTES EACH)
LAST RECORD IS SHORT
C
AN INTEGRAL NUMBER OF ELSI RECORDS (185,344 BYTES EACH)
FILE LABEL (1024 BYTES)
D
20 ELSI IMAGES (9216 BYTES EACH)
SPACER (8 BYTES)
RECORD LABEL (1016 BYTES)
E
48 COLUMNS (192 BYTES EACH)
F
16 ROWS (12 BYTES EACH)
G
6 SECTIONS (2 BYTES EACH)

Figure 1. Tape format used for the recording of ELSI data

the six sections before it proceeds to the next location in the next row. It progresses through all row locations in a column before moving to the next column. Thus, the resulting image array has dimensions of section, row, and column (s,r,c), where the s index cycles most rapidly and the c index most slowly. There is a further complication in that the readout progresses through 16 (not 15) rows in each section and 48 (not 45) columns. The resulting spurious values are referred to as *overclocked pixels* and must be discarded in processing the data. These are apparent in Figure 3, which illustrates the structure of the image array; the overclocked pixels are represented by open squares, while real pixels are shown as solid squares.

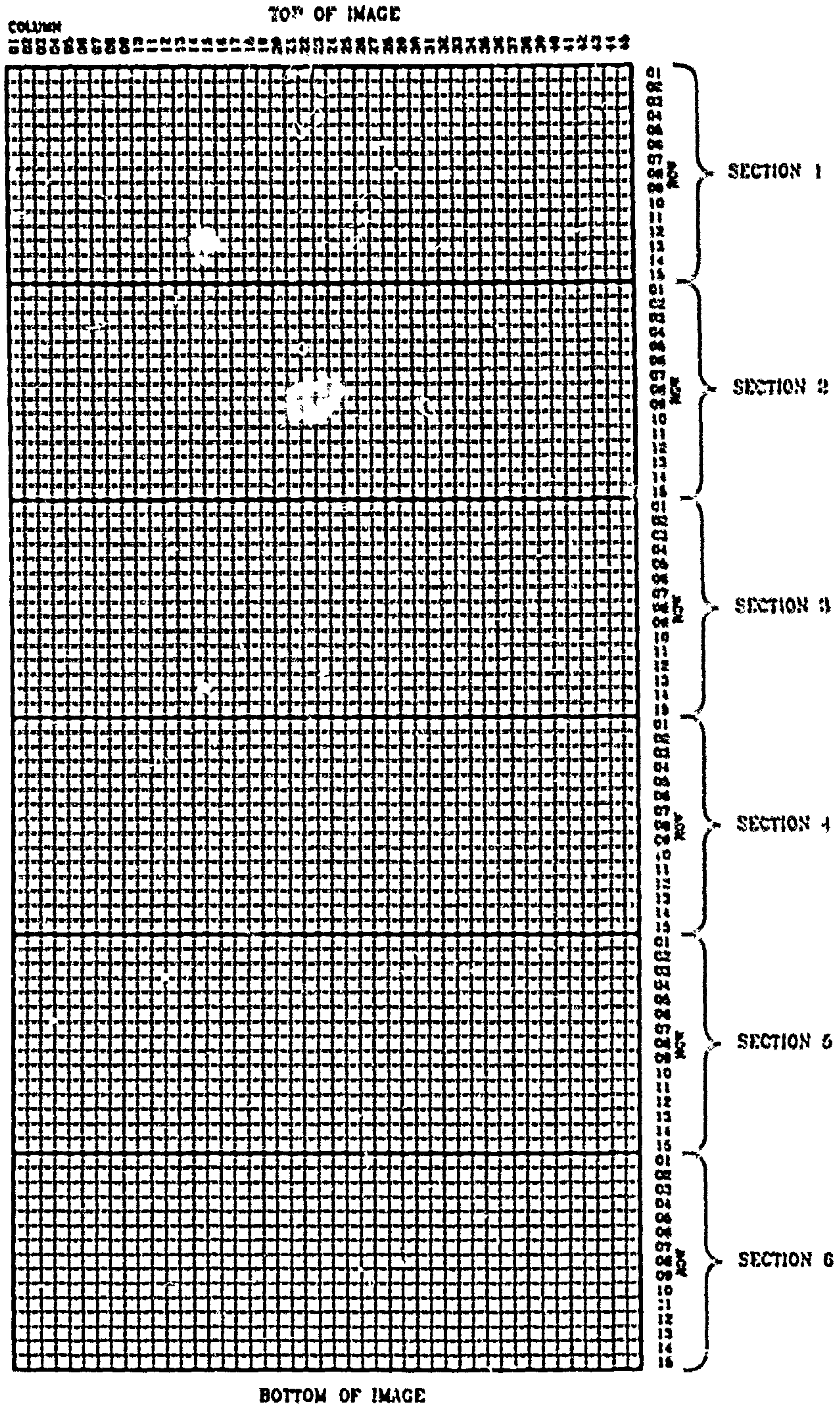


Figure 2. Format of pixels on the ELSI MWIR detector

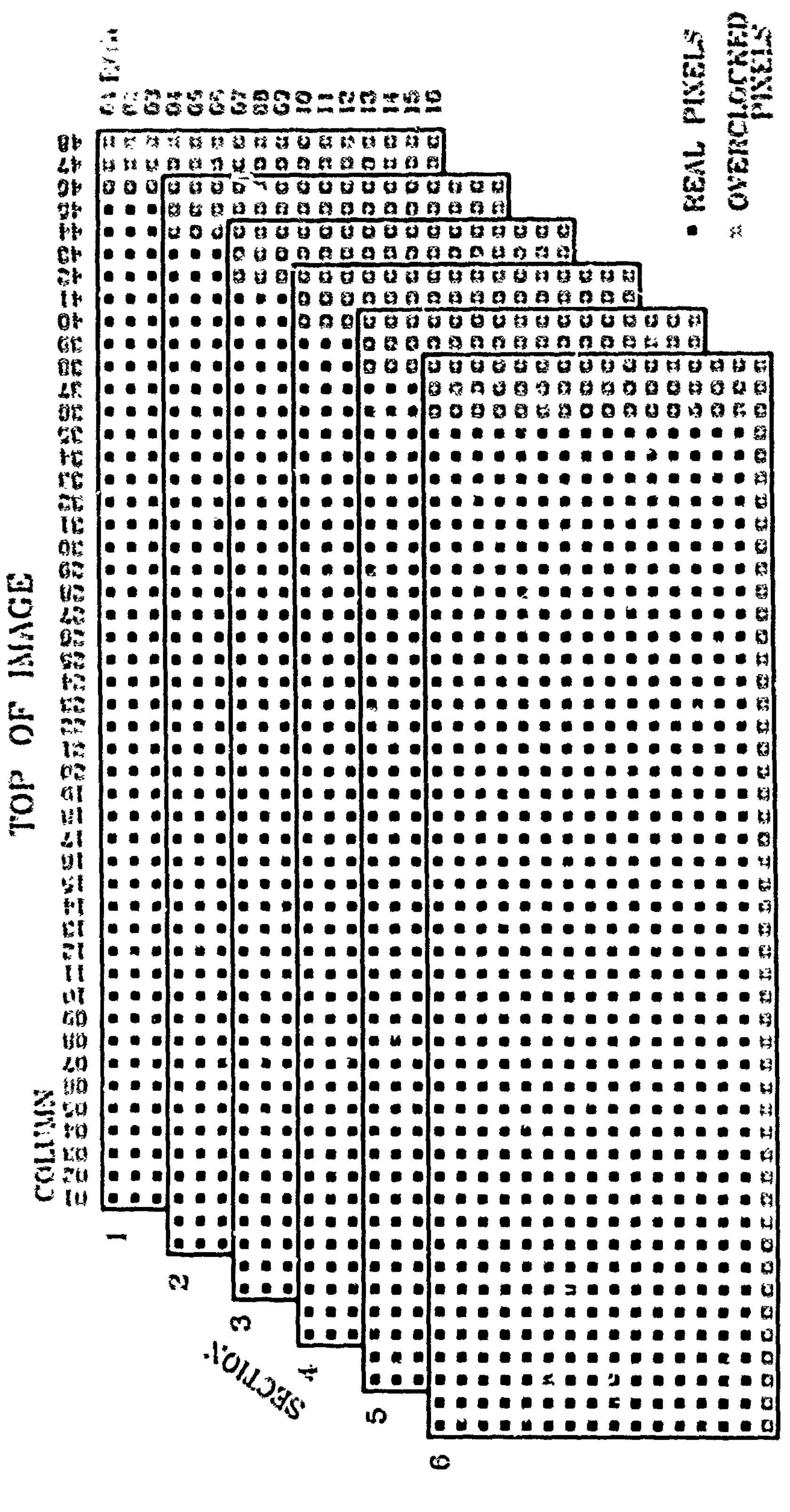


Figure 3. Format of an image data array within an ELSI record

4. EQUIPMENT AND PROCEDURE

Figure 4 shows, as a function of time, the data recorded for orbit 67. Culmination, the time of closest approach of the shuttle to AMOS, is indicated by an arrow at the top. Two tracks of solid bars at the bottom show which detector was in use at a particular time. Above that is shown the one-minute period, just after culmination, for which digital data were requested; it includes a small amount of LWIR data. The data actually received are indicated at the top; they cover the requested period and continue until the tape reel is filled. The division into two files probably reflects a switchover to a new tape unit during the original recording. It provided a convenient starting point for our examination; all of the data discussed below come from file 2.

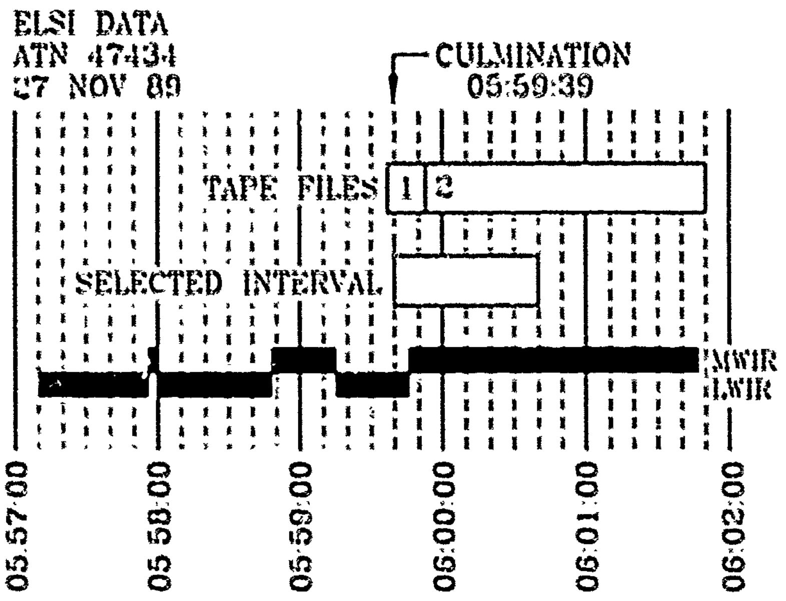


Figure 4. Summary log of the orbit-67 ELSI data used in this work

The equipment used was a Zenith Z-248 desktop system, a PC-AT clone, equipped with dual 22-megabyte Bernoulli drives in addition to 20- and 40-megabyte internal disks. Also included was a Cordata LP300X laser printer equipped with a custom font of 128 grey-scale characters. Except for lack of a tape drive, this system proved nearly ideal for examining these data. Data were transported to this system on a Bernoulli disk. The first six megabytes of tape file 2 were copied from tape to a Microvax user directory on disk and then, via local network, to the Bernoulli cartridge. This data extract comprises a little more than 32 ELSI records. Figure 5 indicates schematically how the data treatment progressed from this point through subsequent steps described below.

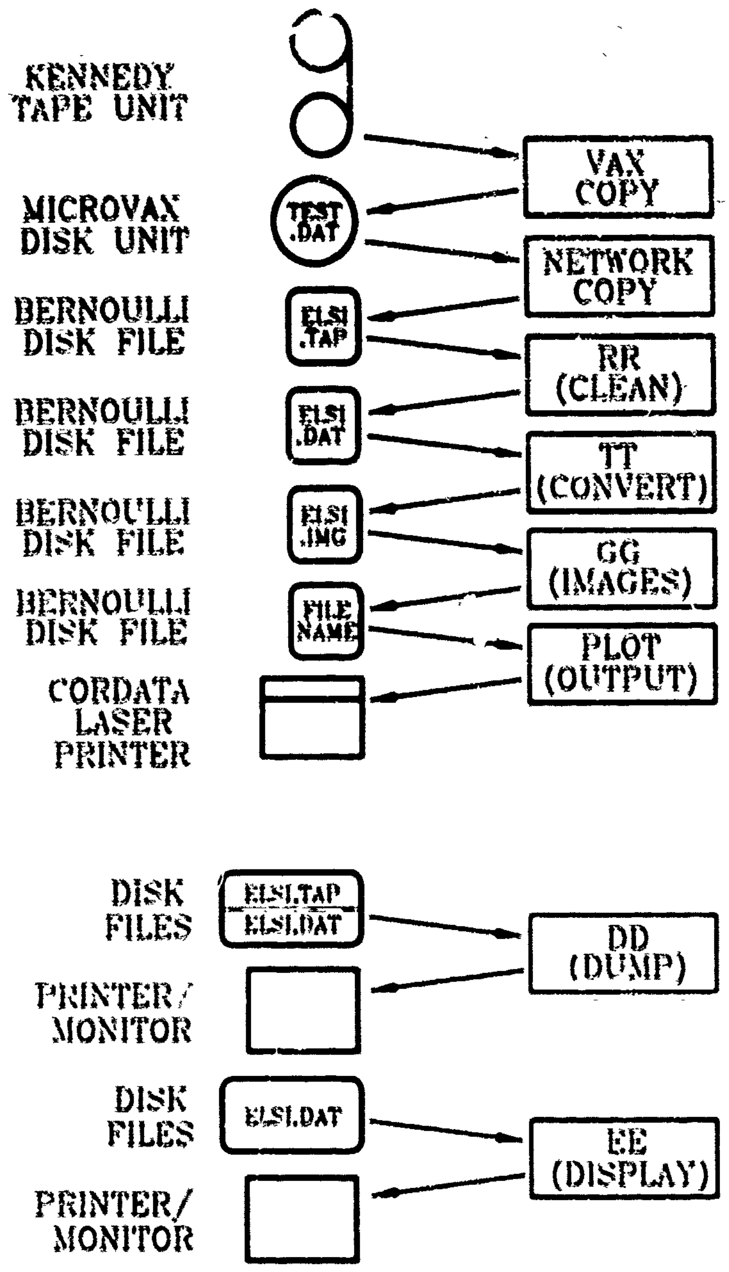


Figure 5. Schematic diagram of the processing procedure

The approach to the problem set four sequential objectives: (1) to discover what was wrong with the ELSI data or the previous efforts to use it; (2) to see whether the data of orbit 67 could be processed to produce images that matched the video data; (3) to transfer whatever method was developed to the LCY facility, applying it first to the orbit-67 data for confirmation; and (4) to use the established technique to process the classified data.

5. EXAMINING THE DATA

Pixel values in the ELSI data originate in a 12-bit A/D converter and are positive integers ranging from zero through 4095; this is true for all twenty of the raw images of an ELSI record. Values are stored as two-byte integers with the four most significant bits set to zero. In practice, the values may be confined mainly to a much narrower range than the full 4096 values allowed. The principal problem with the data was revealed immediately when the distribution of values in a small sample was examined. All values lay between 3456 and 3583 or between 3968 and 4095. What these ranges have in common is that bits 11, 10, 8, and 7 of the 16-bit integer always have the value one; i.e. these bits are "stuck" in the *on* state. Bits are numbered from LSB = 0 through MSB = 15. A closer look also showed that bit 1 was also stuck in the *on* state. With five of twelve bits inoperable, the data are enormously ambiguous. The resulting response curve, plotted in Figure 6, shows that each recorded number represents not the one true value but any one of sixteen possibilities. The scale of this curve does not show the effect of stuck bit #1, which has a data value of only 2. As will be seen later, its effect on an image will be negligibly small, so it is ignored in all of the following discussion.

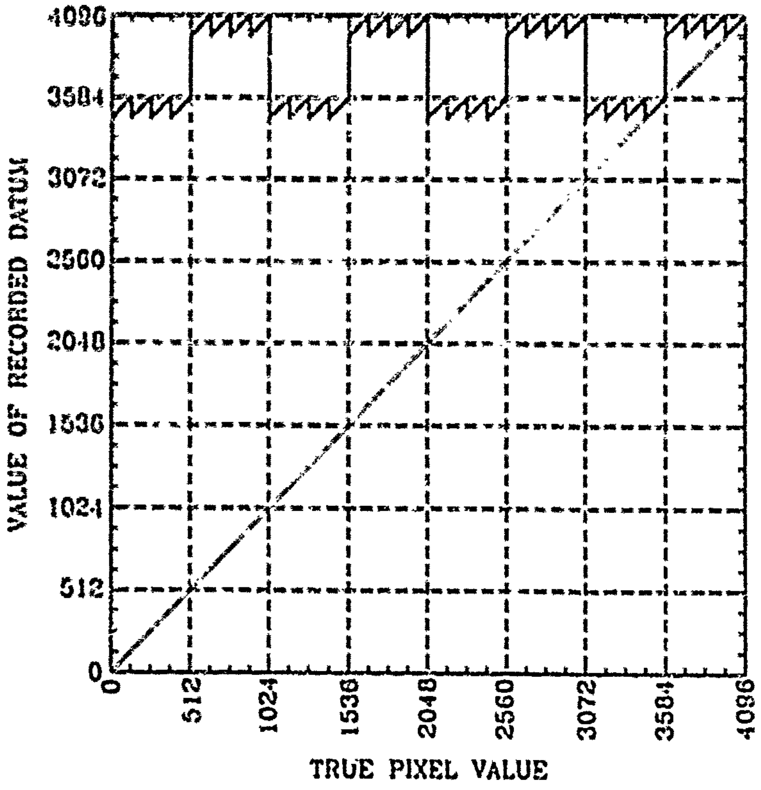


Figure 6. Plot of recorded datum value versus the true pixel value

This discovery was reported to AMOS with a request to check the current status of the instrument. We were subsequently informed that the bits were found to be stuck exactly as reported. We were not told the location and cause of the malfunction, but because the video data, which are stated to be derived by reconversion from the digital, seem to be unaffected by the stuck bits, we suspect the bits are stuck in the recording equipment, not in the data from the telescope.

The next step was to write routines to recognize, read, and manipulate ELSI records, as dictated by the format of Figure 1. One routine was required to clean out extraneous material, because in reading the tape the Microvax copy utility had inserted an end-of-record mark, 0D0A, after the data from each tape record. The next routine operates on each ELSI record to decode time, discard overclocked pixels, reorder real pixels, and write a new data file.

Examining the data with these tools disclosed two more apparent faults. First, one section of the detector, fifth from the top, was completely dead, with all pixels reading the saturation value 4095. This is a relatively minor problem in this case, since the tracked object was kept well within the upper four sections of the detector. Second, time values recorded in successive frames do not advance by the nominal amount of 250 ms (20 frames at a rate of 80 per second). Figure 7 shows the advance of recorded time over a series of 32 frames, revealing that the time interval per frame oscillates irregularly between about 300 and 700 ms, with the average being about 410 ms. It is not known whether this variation is real or the result of inaccurate reading of the clock. It too is a minor problem here, since accurate timing is not required. These faults have not yet been reported to or confirmed by AMOS personnel.

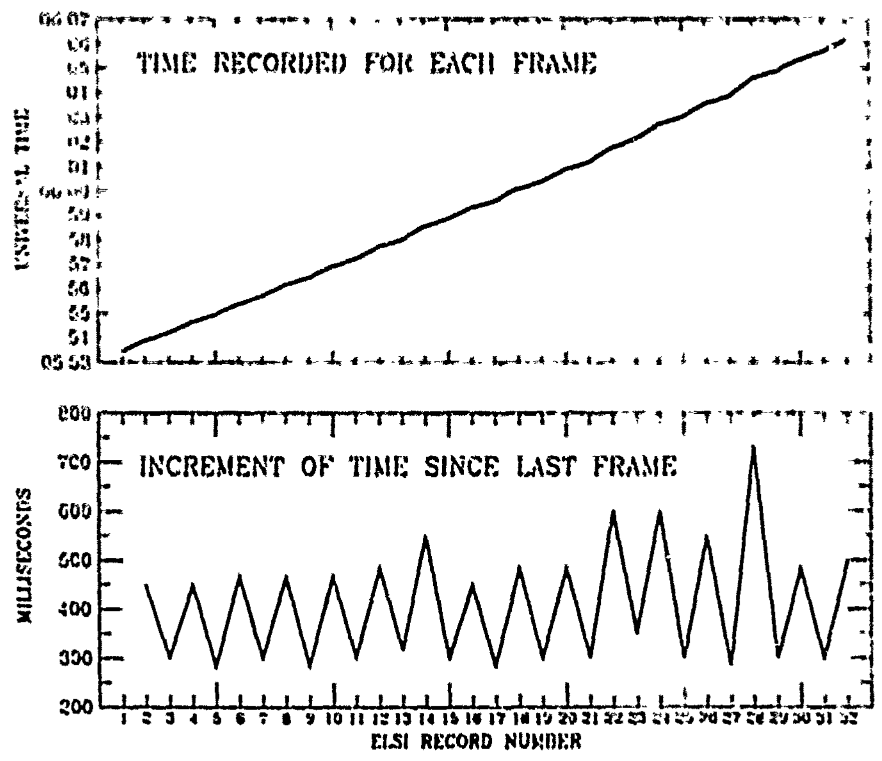


Figure 7. Observations of recorded time versus the record number

A general-purpose image-processing routine was written to permit the production of hardcopy grey-scale prints from the data; this was needed if any method of salvaging the data were to be found. In its latest form, the routine incorporates a number of sequential but repeatable steps, as follows:

a. *Read* the next record, interchange bytes, and convert to integers.

b. *Subtract* each odd background image from the following even image.

c. *Correct* the data using an incorporated algorithm.

d. *Average* over a specified number of successive image pairs.

e. *Process* the composite image, even-minus-odd or odd-minus-even, by finding maximum and minimum data values, setting and adjusting threshold and saturation levels for the grey-scale, and converting pixel values to grey-scale values.

f. *Output* the resulting image to a print file for transfer to the laser printer.

A batch file was used to copy the print file to the laser printer along with the required grey-scale setup and reset commands. With defect-free data, these procedures will produce a good image. The custom GREY3 font relates coverage, or density, of a square pixel to an ASCII character number by a conversion curve that consists of three connecting linear segments that lie about midway between a pure linear and a pure logarithmic scale.

6. CHOOSING A CORRECTION METHOD

A first version of the image processor, with no correction algorithm, was applied to the unaltered data to see how much, if any, of the true image could be seen despite the stuck-bit malfunction. Figure 8 shows the result of applying this normal processing to the first image pair of the first record. The data printed represent image 2 minus image 1, with negative pixel values set to zero. The grey-scale threshold (black) was set at zero, and the saturation (white) was set to the maximum value found in the data.

This image reveals several interesting features. First, the fact that detector section 5 is not operating is evidenced by the black band across the lower half of the image. Second, the location of the true image of the space shuttle is easily discernible as the light area centered near the left edge of section 2, roughly at column 10, row 27. The light-to-medium grey pixels clearly show the shuttle outline as it appears in the videotape. The video image of the shuttle is a vaguely triangular blurry white silhouette against an almost-black background; there is some, but not much, variation within the silhouette. In the digital image roughly half of the pixels within the silhouette have dropped out and appear black. At the location to which the image is deflected, centered roughly at column 10, row 45, there is a collection of bright pixels forming a sort of ghost image where there should be only a completely black area or "hole". A number of other false bright pixels are scattered about the rest of the background.

(on facing page)

Figure 8. A background-subtracted image produced by normal processing

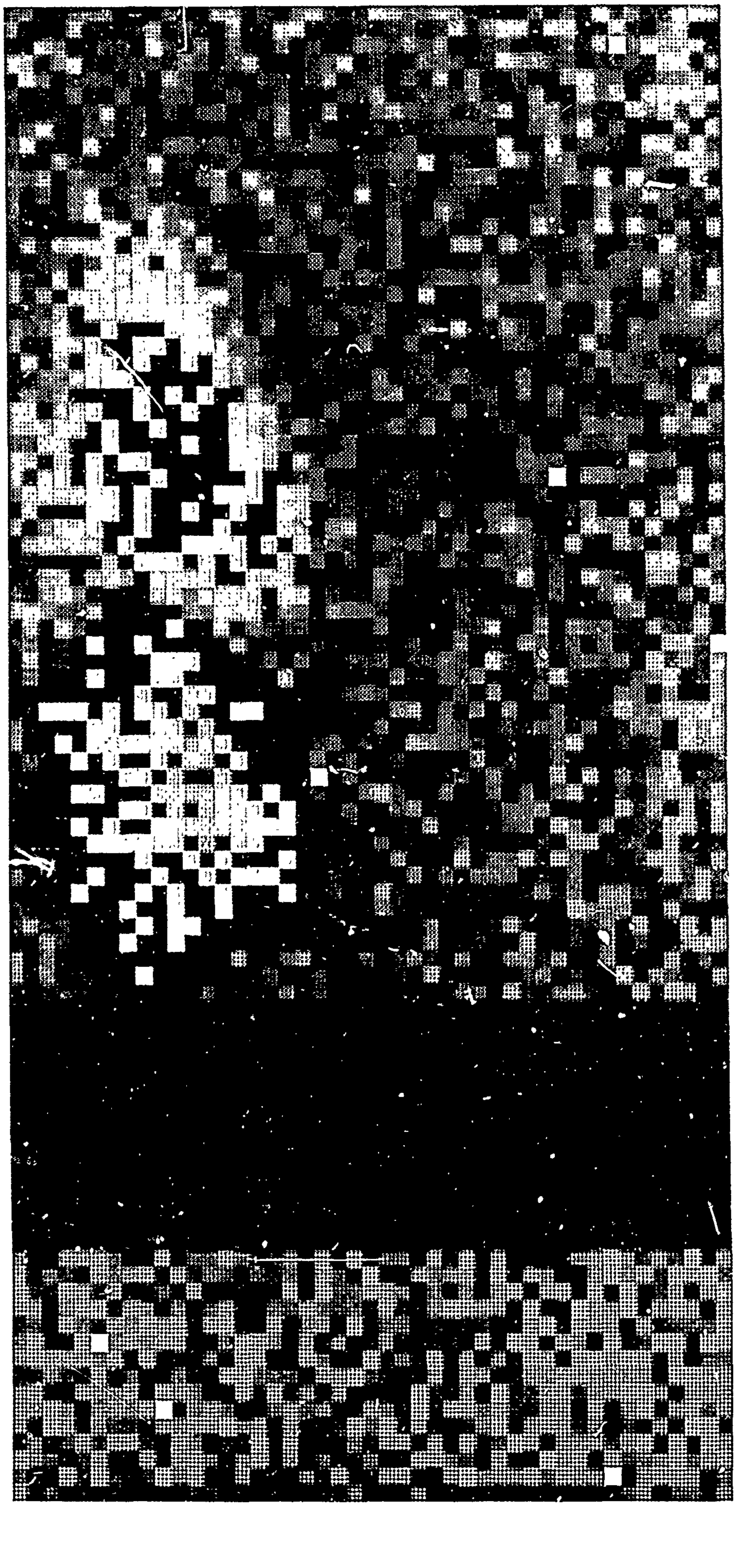


Figure 9 shows image 2 alone, without subtraction of the background image. There is no recognizable structure in the region of the shuttle silhouette. The pixel values show some cohesive appearance; from upper left to lower right the pixels successively assume tones of dark, light, and medium grey in three rather distinct areas. The presence of only two boundaries suggests that no more than two of the larger jumps in the response curve, 384 or 640, are encountered, and only one if the second boundary is a retracing of the first. We therefore expect the distribution of true data values to be confined to a very narrow part of the zero-to-4095 range.

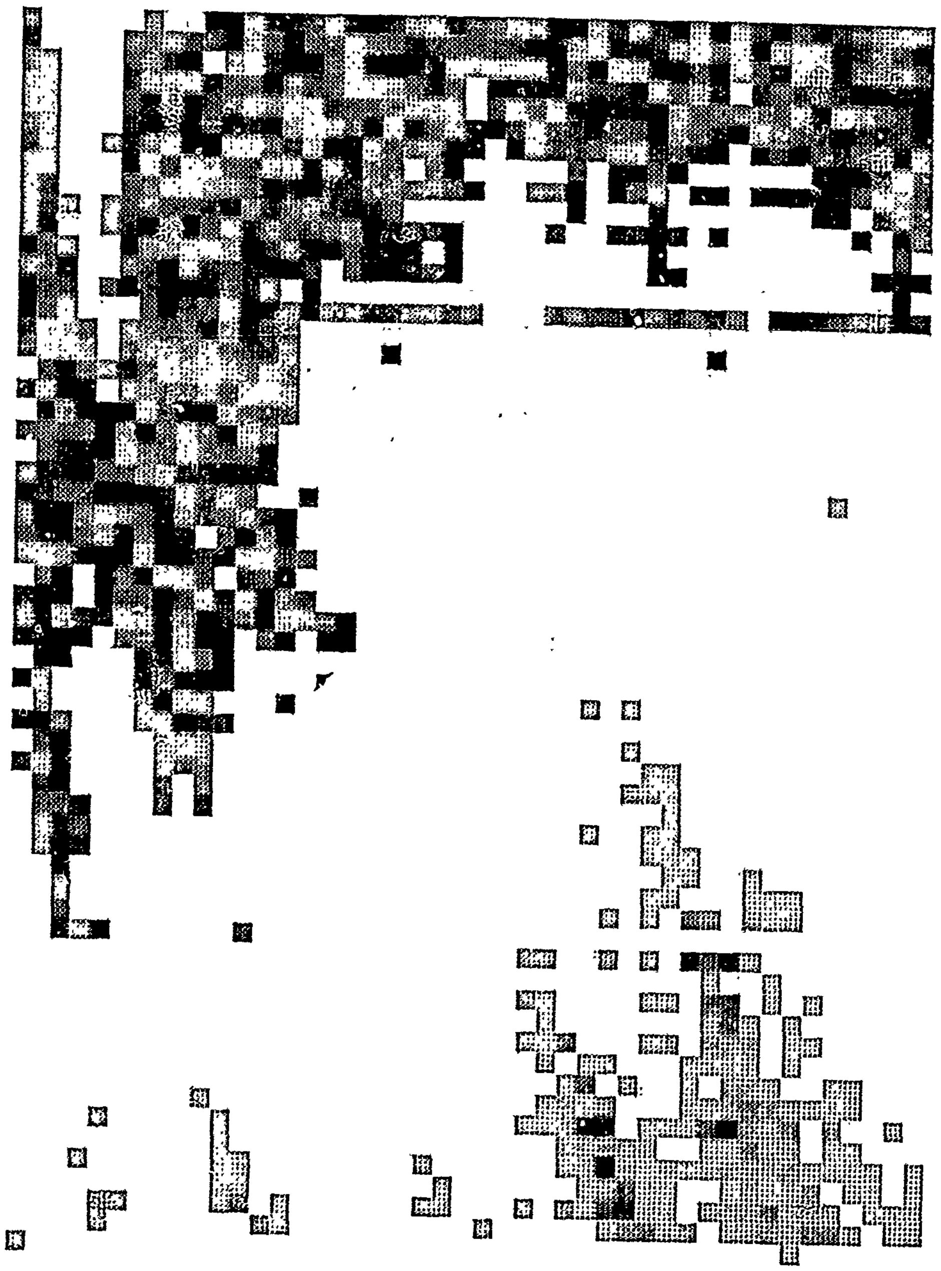
In the background-subtracted image of Figure 8, the abrupt disappearance of grey tones or the appearance of bright spots, as one moves spatially across the image along some line, is easily understood on the basis of the response curve of Figure 6. It occurs when the true value of the even (odd) image crosses one of the discontinuities in the response curve while the odd (even) does not. The dark pixels within the shuttle outline are caused by a negative jump in the even image or a positive jump in the odd image. The bright pixels within the black hole are caused by the reverse condition. There are relatively few false bright pixels, and presumably a similar number of false black pixels, throughout the background because it is rather unlikely that the illumination on such pixels will be much different in the even and odd images and therefore unlikely to straddle a jump in the response curve.

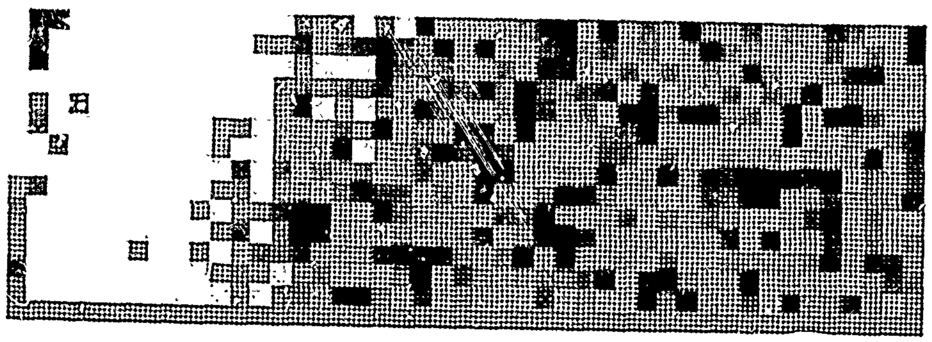
The fact that many of the false pixel values can be recognized by visual examination suggests that an algorithm might be devised to correct the data despite the ugly nature of the response curve. The eye discerns bad pixels because of their sharp jumps from the values of neighboring pixels. The jumps are noticeable because of the relatively low spatial gradient of intensity within the true image. In approaching the shuttle silhouette from any side, we notice that the intensity increases smoothly over several pixels before hitting a jump that makes it false. This suggests that the average spatial gradient may be much less than 128 units per pixel-spacing distance. Since the smallest jump in the response curve is 128 units, these jumps may well be recognizable in going from pixel to pixel in the image. As noted above, the true pixel values seem to be confined to a narrow portion of the full 12-bit range.

The foregoing observations suggest using a very simple and straightforward algorithm for correcting the data. This method takes one starting pixel as a reference point and then moves along multiple tracks across the detector, adjusting the value of each pixel according to whether and how much it has jumped relative to the preceding pixel on the track. The pattern of tracks is a tree with enough branches to cover every pixel. Since any error in identifying a jump will affect every subsequent pixel on a branch, it is well to make the branches as short as possible. It is not really necessary for the starting pixel to have a true value, since the absolute level of the result is not important unless successive images are to be averaged. But in any case, the amount of its error will be apparent from the difference in the resulting distributions of positive and negative values; the black background will not be roughly zero but some value that approximates the size of jump by which the starting pixel was in error, and the magnitudes of the most positive and most negative pixels will differ by twice the size of that jump.

(on facing page)

Figure 9. A single even-numbered image produced by normal processing





7. CONSTRUCTING THE ALGORITHM

For clarity in the following discussion, all results presented will be for the image pair already shown (file 2, record 1, image pair 1), and references to the sample image will mean the background-subtracted image unless stated otherwise. Also, in much of the work to follow, only the upper two-thirds, rows 1 through 60, of the detector will be used. The fifth section is dead, and since the detached sixth section is of little use by itself, it was usually set to zero.

We know that the distribution of values in a single image is confined to the two ranges noted earlier: 3456-3583 and 3968-4095, but this will be different in a background-subtracted image. Figure 10 shows the distribution for the sample case. This and the following figure use logarithmic scales except that zero has been placed at the 0.1 level. Values are less confined because two quantities are involved in the subtraction. We may reasonably guess that many of the values below 128 are true, while most of the two higher groups are false. We also note that over most of the image, where there is no tracked object, both the even- and odd-image values of a pixel may be grossly affected by the stuck-bit malfunction while their difference remains correct.)

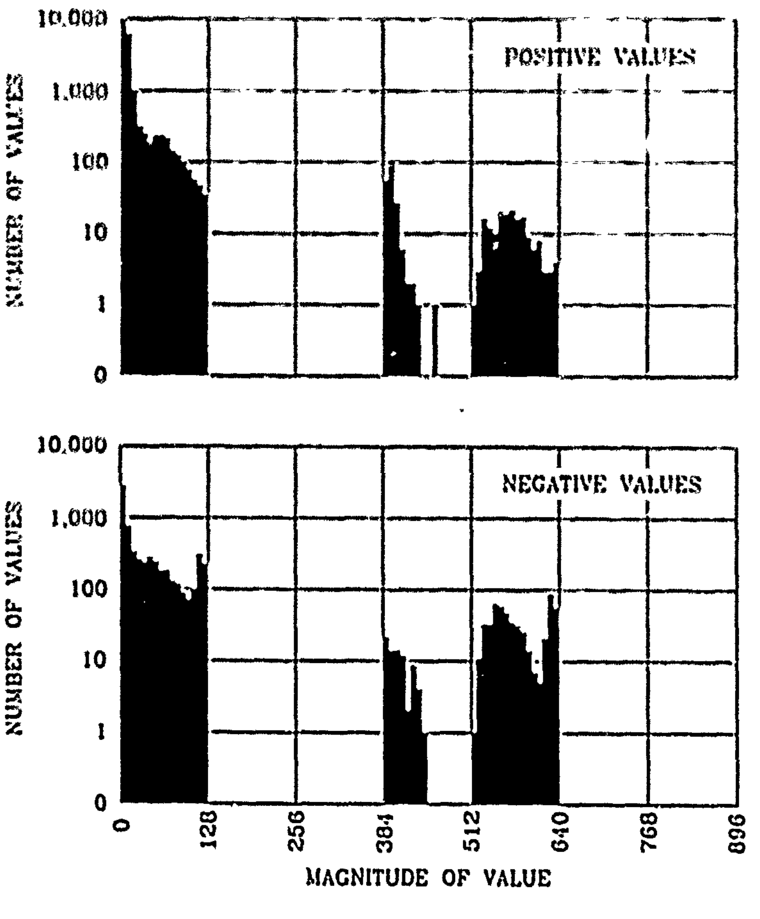


Figure 10. Distribution of pixel values in an uncorrected image

The distribution of the actual pixel-to-pixel jumps in an image is most important to know. Figure 11 shows the distribution of jump sizes encountered in scanning left to right along all of the first 60 rows of an image. Jumps having magnitudes of 1, 3, and 5, in units of 128, are expected, since they occur directly in the response curve; the pixel needs to jump in only one image of the subtracted pair. We also see many jumps of magnitude 4. In this case the pixel must jump in both images; for example (+3)-(-1) or (+5)-(+1). Both of these combinations are likely, as can be seen by juxtaposing the curve of Figure 6 and its inverse; a difference of only 128 in the true values of even and odd images can cause these jump combinations to occur if the spatial gradient has the same direction in both images, as is usually the case. Conversely, jumps of magnitude 2 and 6 are almost never seen. These also require jumps in both images; for example (+3)-(+1) or (+5)-(-1). Such jumps are also possible with the same even-odd difference of 128, but the spatial gradients in the even and odd images must be in opposite directions. Since this is very unlikely, these jumps are very rare.

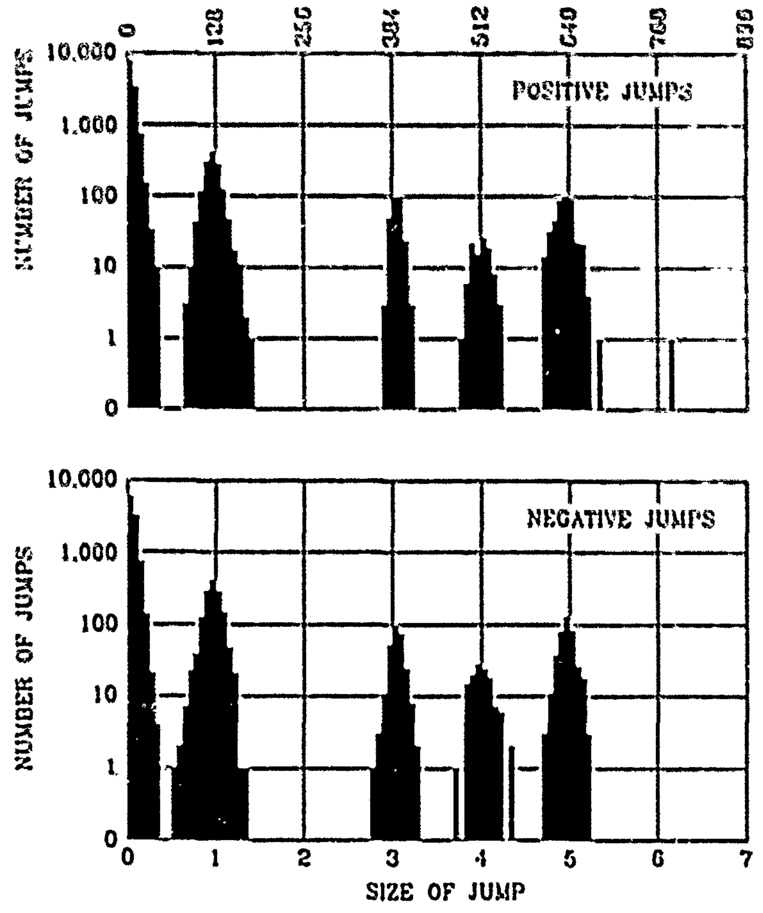


Figure 11. Distribution of pixel-to-pixel jumps in an uncorrected image

Clearly, the important feature of this distribution that makes correction possible is the clear separation of the magnitude-0 and magnitude-1 groups. This reveals that the pixel-to-pixel variation in the true image does not exceed about 48, while a stuck-bit jump over a discontinuity of 128 never results in a jump of less than 64. Similarly, jumps of magnitude 3, 4, or 5 are equally unambiguous. This implies that an algorithm based on jump size alone will suffice, and there will be no need to invoke such additional criteria as the local trend of the spatial gradient, etc.

The correction algorithm, therefore, was constructed to work as follows. The starting location is taken to be the top of the middle column (column 23, row 1) and the first track runs down column 23 to row 60. From the central column of corrected pixels, each subsequent track runs along a row from the central column to the right or left edge, so the complete pattern comprises 121 tracks, with only the initial track covering more than 23 locations. At each step along the track the observed jump to the current pixel is classified as to the discontinuity included, an integral number between -6 and +6. This number is added to the cumulative total from the beginning of the track to find the amount by which the current pixel should be adjusted. The cumulative adjustment for each pixel is stored in a separate array, and the pixels are adjusted after completing all tracks.

The threshold levels used to determine jump size, that is, the location of boundaries between the groups in Figure 11, were allowed to be individually adjusted, but in practice setting them all to the appropriate multiple of 64 was found to be adequate.

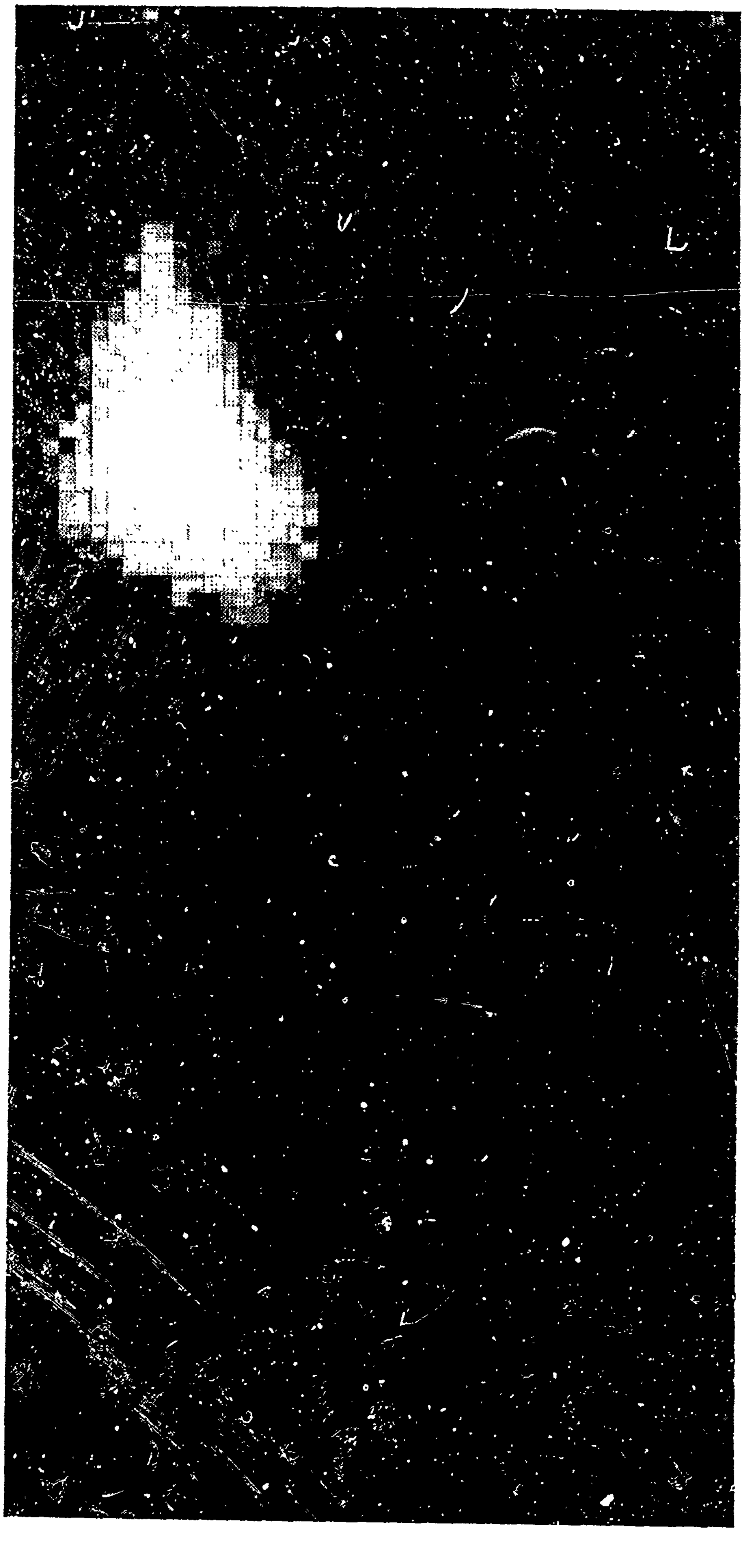
It may be useful to note that we first tried to use a similar algorithm applied separately to the odd and even images of a pair. It was thought that this might prove best, so that legitimate variations in both images would not be added, possibly making their sum large enough to be confused with a magnitude-1 jump. This was a mistake. That effect was outweighed by effect of inherent pixel-to-pixel variations in response. The latter, which creates the need for flat-fielding techniques in the first place, proved to be larger than pixel-to-pixel variations in the image. They were, in fact, so large that they could not be distinguished from magnitude-1 jumps, and the algorithm failed.

8. DESCRIPTION OF RESULTS

The algorithm proved completely successful in correcting the orbit-67 data used in its development. Figure 12 shows the fully corrected sample image. In this case, grey-scale saturation (white) was set to correspond with the brightest pixel, and grey-scale threshold (black) was set somewhat above zero, as is usually done, to suppress the empty-sky background in addition to eliminating the deflected image due to nodding. Figure 13 show [illegible] he same image without the sky background suppressed (grey-scale threshold at zero) [illegible] background has a fairly uniform mottled appearance, and the black hole of the deflect [illegible] ble.

(on facing page)

Figure 12. The fully corrected image with sky background suppressed



Some testing of the results was done, but no significantly different data were available for a rigorous evaluation. As long as the jump distribution shows complete separation between groups, the correction should be perfect. It is not an approximation, so even the most subtle features in the true image will be preserved. Moreover, if an error in detecting and classifying a jump were to occur, it would expose itself by producing a very noticeable defect in the resulting image, a defect propagating along the remainder of the correction track. This was demonstrated in testing by deliberately forcing an error. We expect that such a defect would still be apparent even if the image were too active for the correction to succeed.

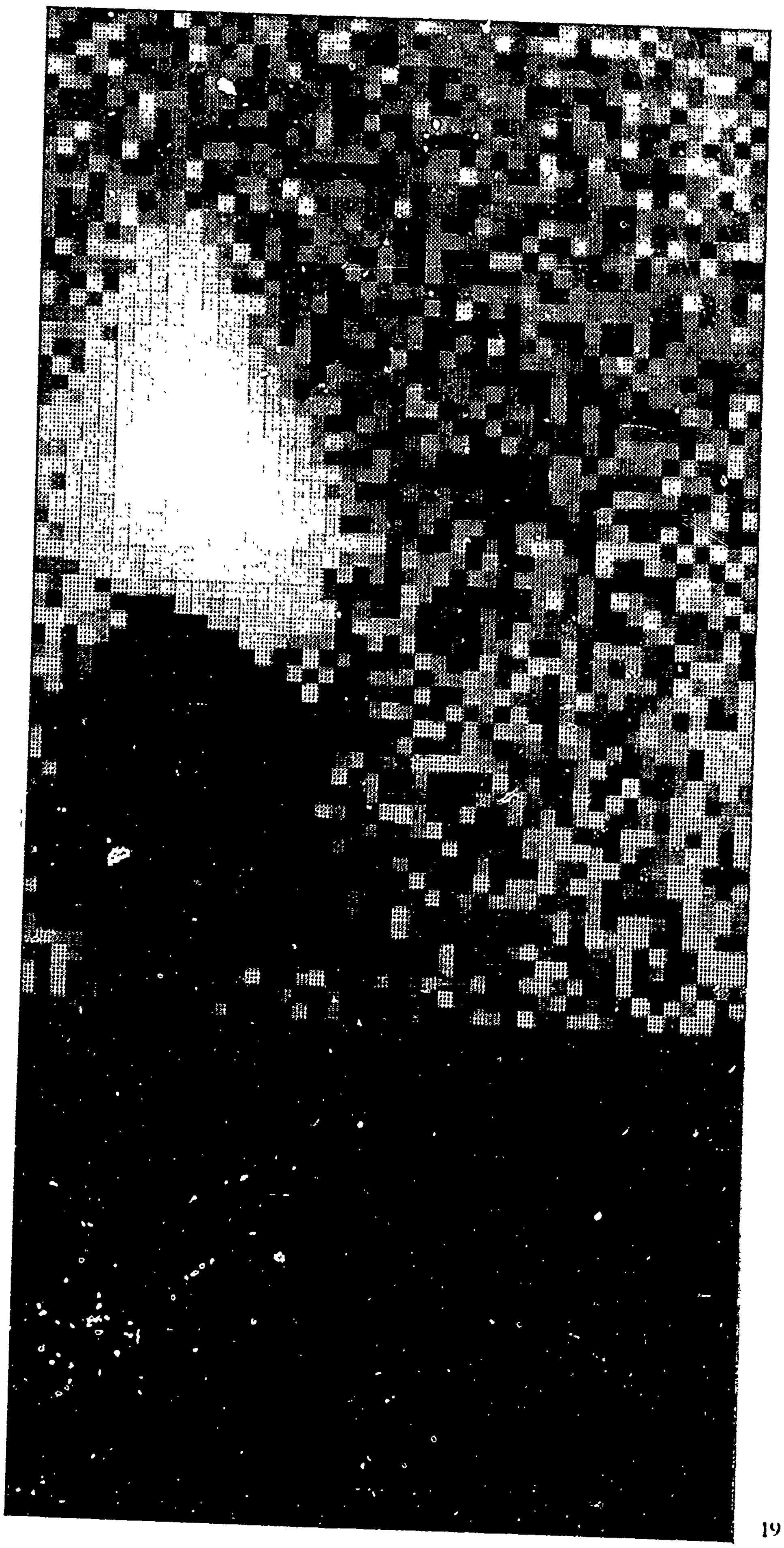
The successful correction of these data reveals one more condition that may be another significant fault in the instrument. The tracked object seen in Figures 12 and 13 is the space shuttle, which is only 40 meters in length, at a distance from the telecope of more than 200 miles. Still, in Figure 13 we can clearly see that the black hole of the deflected image has obscured a small part of the desired image. If the object were either larger or closer, more of the image would be obscured. Since the background subtraction works very well with the current amount of image deflection, it should be possible and useful to increase the deflection, moving the black hole farther out of the way. We have not yet enquired about the practicality of doing so.

At the time of this writing, the algorithm has been transferred to LCY for processing of the classified data. A preliminary test on their system, using the same unclassified data, appears to have produced the same result, but because of security-classification difficulties that system currently lacks a grey-scale output, a problem that is currently being addressed.

FORTRAN source listings for most of the software written for this work are reproduced in the Appendix. Only the most recent versions are included. Some of the results presented above were produced with earlier versions.

(on facing page)

Figure 13. The fully corrected image with grey-scale threshold at zero



Appendix

SOFTWARE SOURCE LISTINGS

FORTRAN source code for the following routines are listed below, in the order shown here. The code is written for an IBM PC-compatible computer running under DOS.

DD	A simple dump of the data file, in 512-byte blocks, to the monitor or a disk file.
RR	A routine to remove spurious end-of-record marks inserted after each tape record by the tape-copy utility.
EE	An ELSI-record reader to extract and examine one record (spacer, images, and label) at a time.
TT	An image translator to read the raw-data file, extract and decode time, discard overclocked pixels, reorder the array, and write a new data file.
DS	A routine to compute distributions of pixel values and pixel-to-pixel jumps.
GG	An image processor to read, subtract, correct, average, process, and output the results from one ELSI record.

DD ROUTINE (DATA DUMP):

```
      PROGRAM DD
*---------ELSI data dump to examine unprocessed data.
*---------variables, etc
      CHARACTER*1 A(512)
*---------display program identifier
      WRITE(*,800)
  800 FORMAT(////////,
     + 20X,'┌──────────────────────────────────┐',/,
     + 20X,'│       ELSI DATA-FILE DUMP (DD)   │',/,
     + 20X,'│                                  │',/,
     + 20X,'│    Dumps an ELSI data file in    │',/,
     + 20X,'│    512-byte blocks.  Output is   │',/,
     + 20X,'│    to the screen and also to a   │',/,
     + 20X,'│    file or a printer or both.    │',/,
     + 20X,'│    Reads the input file ELSI.DAT │',/,
     + 20X,'│    Writes to a file PRINTOUT.DOC,│',/,
     + 20X,'│    both of which must exist in   │',/,
     + 20X,'│    the current default directory.│',/,
     + 20X,'│    Writes to the PRN printer.    │',/,
     + 20X,'│                                  │',/,
     + 20X,'│    Press return to begin reading │',/,
     + 20X,'└──────────────────────────────────┘',/,
     + 4(/))
```

```
      READ(*,700)Z
  700 FORMAT(A1)
*---------open files
      OPEN(50,FILE='ELSI.DAT',FORM='BINARY',
     * ACCESS='SEQUENTIAL',IOSTAT=IOC,STATUS='OLD')
      OPEN(60,FILE='PRN')
      OPEN(70,FILE='PRINTOUT.DOC',FORM='FORMATTED',
     * ACCESS='SEQUENTIAL',STATUS='OLD')
*---------discard file label
      READ(50)A
      WRITE(*,802)A
  802 FORMAT(/,
     * ' TAPE-FILE LABEL ',63(1H=),
     * 8(6X,64A1,/),/,
     * 1X,79(1H=),//,
     * ' Press return to continue ...')
      READ(50)A
      READ(*,700)Z
*---------start dumping blocks
      DO 40 IREC=1,40
      NEXT=1
      DO 30 IBLK=1,362
      READ(50)A
      IF(IOC.NE.0)GOTO 45
      IF(NEXT.GT.IBLK)GOTO 30
      IF(NEXT.EQ.IBLK)GOTO 15
      WRITE(*,804)
  804 FORMAT(1X,'Skip to block N?  N= ',\)
      READ(*,704)IDUM
  704 FORMAT(BN,I6)
      IF(IDUM.GT.IBLK)NEXT=IDUM
      IF(IDUM.LE.IBLK)NEXT=IBLK
      IF(IDUM.LT.0)NEXT=363
*---------display block
      IF(NEXT.GT.IBLK)GOTO 30
   15 WRITE(*,806)IREC,IBLK
  806 FORMAT(1X,'ELSI RECORD',I3,6X,'BLOCK',I4)
      WRITE(*,808)A
  808 FORMAT(16(4(2X,8Z2),/))
      WRITE(*,810)
  810 FORMAT(1X,'Copy to Printer, File, Both, ',
     * 'None? (P/F/B/N) (N) ',\)
      READ(*,700)Z
      IPRT=0
      IFIL=0
      IF(Z.EQ.'p')Z='P'
      IF(Z.EQ.'f')Z='F'
      IF(Z.EQ.'b')Z='B'
      IF(Z.EQ.'P'.OR.Z.EQ.'B')WRITE(60,806)IREC,IBLK
      IF(Z.EQ.'P'.OR.Z.EQ.'B')WRITE(60,808)A
      IF(Z.EQ.'F'.OR.Z.EQ.'B')WRITE(70,806)IREC,IBLK
      IF(Z.EQ.'F'.OR.Z.EQ.'B')WRITE(70,808)A
   30 CONTINUE
*     -------end of blocks in record
   32 WRITE(*,825)
  825 FORMAT(//,
     * ' Read another ELSI record (Y/N)? ',\)
      READ(*,700)Z
      IF(Z.EQ.'Y'.OR.Z.EQ.'y')GOTO 40
      IF(Z.EQ.'N'.OR.Z.EQ.'n')GOTO 50
      GOTO 32
   40 CONTINUE
*     -------end of records
*---------abnormal read
   45 IF(IOC.GT.0)GOTO 47
      WRITE(*,830)
  830 FORMAT(' EXIT:  End-of-file encountered')
      GOTO 50
   47 WRITE(*,832)
  832 FORMAT(' ABORT: Read error in ELSI.DAT')
*---------exit
   50 CONTINUE
      CLOSE(50)
      CLOSE(70)
      WRITE(*,850)
  850 FORMAT(///,
     * 20X,'┌──────────────────────────────────┐'/,
     * 20X,'│   END OF ELSI DATA-FILE DUMP (DD)  │'/,
     * 20X,'└──────────────────────────────────┘'/,
     * 17(/))
      END
```

RR ROUTINE (EOR REMOVER):

```
*********
*     PROGRAM RR
*       Record-mark remover for ELSI data file.  The
*       Bernoulli-disk sample of ELSI data was copied
*       from the AMCS tape on the FH microvax.  This
*       process inserted a CRLF (0D0A) at the end of
*       each tape record of 16,384 bytes, so there are
*       about 22 or 24 extra (unpredictable) bytes in
*       each ELSI record.  This routine copies the data
*       to a new file, discarding end-of-record marks.
*
*---------variables
      CHARACTER*1  A(16386),B(16384)
      INTEGER*1    C(2)
      INTEGER*2    MARK,CC
      EQUIVALENCE  (A(1),B(1))
      EQUIVALENCE  (A(16385),C(1)),(C(1),CC)
*---------constants
      MARK=2573
*---------display program identifier
      WRITE(*,820)
  820 FORMAT(////////,
     + 28X,'┌──────────────────────────────────┐'/,
     + 28X,'│                                  │'/,
     + 28X,'│           PROGRAM RR             │'/,
     + 28X,'│   End-of-record mark remover     │'/,
     + 28X,'│  for ELSI data copied from tape  │'/,
     + 28X,'│                                  │'/,
     + 28X,'│  Requires input file:  INPUT.CAT │'/,
     + 28X,'│  Requires output file:  ELSI.CAT │'/,
     + 28X,'│                                  │'/,
     + 28X,'│   Press ENTER to continue ...    │'/,
     + 28X,'└──────────────────────────────────┘'/,
     + 1X(7))
      READ(*,720)Z
  720 FORMAT(A1)
*---------open files
      OPEN(90,FILE='INPUT.CAT',FORM='BINARY',
     + ACCESS='SEQUENTIAL',IOSTAT=IO90,STATUS='OLD')
      OPEN(80,FILE='ELSI.CAT',FORM='BINARY',
     + ACCESS='SEQUENTIAL',IOSTAT=IO80,STATUS='OLD')
*---------read old file and write new file
      WRITE(*,802)
  802 FORMAT(1X,
     + 'Enter number of records to process: ',\)
      READ(*,701)ISTOP
  701 FORMAT(BN,I6)
      IEOF=0
      DO 40 IBLK=1,ISTOP
*---------read old record
   10 READ(90)A
      IF(IO90.GT.0)GOTO 98
      IF(IO90.LT.0)IEOF=1
      IF(CC.NE.MARK.AND.IEOF.NE.1)GOTO 50
      WRITE(*,810)IBLK
  810 FORMAT(1X,'Writing block',I4)
      WRITE(80)B
      IF(IO80.NE.0)GOTO 98
      IF(IEOF.EQ.1)GOTO 60
   40 CONTINUE
      WRITE(*,840)
  840 FORMAT(1X,'END: File exceeds 1000 records')
      GOTO 99
*---------missing end-of-record
   50 CONTINUE
      WRITE(*,850)IBLK
  850 FORMAT(1X,'Missing end-of-record mark',/,
     + 1X,'Mark for block ',I4,' reads ',Z4)
      READ(*,720)Z
      GOTO 99
*---------end of file
   60 WRITE(*,860)IBLK
  860 FORMAT(1X,'End of file in block',I5)
      GOTO 99
*---------abnormal read or write
   98 WRITE(*,898)IO90,IO80
  898 FORMAT(1X,'Diskfile read/write error',/,
     + 1X,'IO90=',I3,6X,'IO80=',I3)
*---------normal exit
   99 CONTINUE
      CLOSE(90)
      CLOSE(80)
      WRITE(*,899)
  899 FORMAT(/,
     + 28X,'┌──────────────────────────────────┐'/,
     + 28X,'│        END OF PROGRAM  RR        │'/,
     + 28X,'└──────────────────────────────────┘'/)
      END
```

EE ROUTINE (ELSI-RECORD READER):

```
      PROGRAM EE
*---------Program to read a file ELSI.DAT that must be in
*         the default subdirectory.  ELSI.DAT is a copy
*         of Tape File 2 from the original AMPS tape for
*         ATN 47434.  The file is 5,937,276 bytes (about
*         11,700 512-byte VAX blocks), representing about
*         eight seconds of data (one sec = 741,376 bytes)
*
*
*---------variables, etc
      CHARACTER*1   A(1024),B(8),C(1816),Z
      CHARACTER*1   TS(4),TXS(4),TM(2),TXM(2)
      CHARACTER*1   D(12,16,48,20)
      INTEGER*2     ID(6,16,48,20),IT2
      INTEGER*4     IT0,IT1,IT,IST,KT(4)
      EQUIVALENCE   (A(1),B(1)),(A(9),C(1))
      EQUIVALENCE   (D(1),ID(1))
      EQUIVALENCE   (C(21),TXS(1)),(C(25),TXM(1))
      EQUIVALENCE   (IT1,TS(1)),(IT2,TM(1))
*---------define constants
*         (0000 hrs 27 Nov 1989 = 628,128,000)
      IT0=628128000
*---------display program identifier
      WRITE(*,800)
  800 FORMAT(////////,
     + 20X,'┌──────────────────────────────────────┐',/,
     + 20X,'│       ELSI DATA-TAPE READER (EE)      │',/,
     + 20X,'│                                      │',/,
     + 20X,'│  This routine reads from the file    │',/,
     + 20X,'│  ELSI.DAT in the default drive.      │',/,
     + 20X,'│                                      │',/,
     + 20X,'│  Press return to begin reading ...   │',/,
     + 20X,'└──────────────────────────────────────┘',/,
     + 12(/))
      READ(*,700)Z
  700 FORMAT(A1)
*---------open input file
      OPEN(50,FILE='ELSI.DAT',FORM='BINARY',
     + ACCESS='SEQUENTIAL',IOSTAT=IST,STATUS='OLD')
*---------read and display file label
      READ(50)A
      WRITE(*,802)A
  802 FORMAT(/,
     + 1X,'TAPE-FILE LABEL: ',/,79(1H-),/,
     + 16(6X,64A1,/),
     + 1X,79(1H-),//,
     + ' Press return to continue ...')
      READ(*,700)Z
*---------process 32 ELSI records (number in file)
*         Note: (5936544-1024 bytes)/185344 = 32.34
      DO 40 IREC=1,32
*---------read a record
      WRITE(*,803)IREC
  803 FORMAT(/,
     + 1X,'Reading in record',I3,' ...',/)
      READ(50)B
      IF(IST.NE.0)GOTO 45
      READ(50)D
      IF(IST.NE.0)GOTO 45
      READ(50)C
      IF(IST.NE.0)GOTO 45
*---------decode time
      DO 3 I=1,4
      IF(I.LE.2)TM(I)=TXM(3-I)
    3 TS(I)=TXS(5-I)
      IT=IT1-IT0
      KT(1)=IT/3600
      IT=MOD(IT,3600)
      KT(2)=IT/60
      IT=MOD(IT,60)
      KT(3)=IT
      KT(4)=IT2
*---------display 8-byte spacer and time
      WRITE(*,810)B,KT
  810 FORMAT(
     + 1X,'EIGHT-BYTE SPACER:    ',
     + 8(Z2,1X),/,
     + 1X,'ELSI-RECORD TIME:     ',
     + I2,':',I2,':',I2,'.',I3, ' UT',/)
*---------display 20 images in hex
      WRITE(*,815)
  815 FORMAT(
     + 1X,'Display hex dump of images (Y/N)? [N] ',\)
      READ(*,700)Z
      IF(Z.NE.'Y'.AND.Z.NE.'y')GOTO 22
      DO 20 IMAG=1,20
      WRITE(*,816)IMAG
 816  FORMAT(/,
     * 1X,'IMAGE NUMBER ',I2,/,
     * 1X,'Enter...D: Display this image',/,
     * 1X,'          S: Skip to next image',/,
     * 1X,'          R: Read the next record',/,
     * 1X,'          E: Exit program ........ (D) ',\)
   5  READ(*,700)Z
      IF(Z.EQ.'S'.OR.Z.EQ.'s')GOTO 20
      IF(Z.EQ.'R'.OR.Z.EQ.'r')GOTO 40
      IF(Z.EQ.'E'.OR.Z.EQ.'e')GOTO 99
   6  DO 18 ICOL=1,48
      IF(ISKP.EQ.1)GOTO 17
      WRITE(*,818)IREC,IMAG,ICOL
 818  FORMAT(/,
     * 1X,'RECORD ',I3,
     * 5X,'IMAGE ',I2,
     * 5X,'COLUMN ',I2,
     * 5X,'6 SECTIONS/ROW)')
      DO 16 IROW=1,16
      WRITE(*,820)IROW,
     * (D(ISCT,IROW,ICOL,IMAG),ISCT=1,12)
 820  FORMAT(1X,'ROW ',I2,8X,6(3X,2Z2))
  16  CONTINUE
*     ---------end rows
      WRITE(*,821)
 821  FORMAT(
     * 1X,'Skip column/Next image/Read record/Exit',
     * 'Continue: (C) ',\)
  17  READ(*,700)Z
      ISKP=0
      IF(Z.EQ.'S'.OR.Z.EQ.'s')ISKP=1
      IF(Z.EQ.'N'.OR.Z.EQ.'n')GOTO 20
      IF(Z.EQ.'R'.OR.Z.EQ.'r')GOTO 40
      IF(Z.EQ.'E'.OR.Z.EQ.'e')GOTO 99
  18  CONTINUE
*     ---------end columns
      WRITE(*,822)
 822  FORMAT(1X,'Begin next image ...',/,
     * 1X,'Skip image/Continue/Exit: (C) ',\)
  19  READ(*,700)Z
      JSKP=0
      IF(Z.EQ.'S'.OR.Z.EQ.'s')JSKP=1
      IF(Z.EQ.'E'.OR.Z.EQ.'e')GOTO 99
  20  CONTINUE
*     ---------end image
  22  CONTINUE
*---------output a modified ELSI record
*               not implemented
*---------optional exit
      WRITE(*,825)
 825  FORMAT(/,
     * 1X,'Read another ELSI record (Y/N)? (Y) ',\)
      READ(*,700)Z
      IF(Z.EQ.'N'.OR.Z.EQ.'n')GOTO 42
  40  CONTINUE
*---------close files
  42  CONTINUE
*     file-write code not present
      GOTO 99
*---------end of file or read error
  45  IF(IST.GT.0)GOTO 47
      WRITE(*,830)
 830  FORMAT(' EXIT:  End-of-file encountered')
      GOTO 42
  47  WRITE(*,832)
 832  FORMAT(' ABORT: Read error in ELSI.DAT')
*---------normal exit
  99  CONTINUE
      CLOSE(90)
      WRITE(*,899)
 899  FORMAT(//,
     + 20X,'┌─────────────────────────────────┐',/,
     + 20X,'│   END OF ELSI DATA READER (EE)   │',/,
     + 20X,'└─────────────────────────────────┘',/)
      END
```

TT ROUTINE (IMAGE TRANSLATOR):

```
      PROGRAM TT
*--------This routine reads in logical records from
*        a standard ELSI data tape and writes out new
*        logical records that contain only the time, a
*        record identifier, and the twenty images.  The
*        pixel values are reordered to a two-dimensional
*        array of 45 columns by 90 rows (from the tape
*        three dimensional array of 48 columns by 16
*        lines by 6 sections).  The overclocked pixels
*        (line 16 of each section and all of columns 46,
*        47, and 48) are discarded during reordering.
*        The input record length is 185344 bytes (181K).
*        The output record length is 162816 bytes (159K)
*        comprising an 816-byte header and 20 8100-byte
*        images (816 + 162000 = 162816 bytes = 159K).
*
*
*--------variables, etc
      CHARACTER*1    A(1024),B(8),C(1016),Z
      CHARACTER*1    TS(4),TXS(4),TM(2),TXM(2)
      CHARACTER*1    D(12,16,48,20),G(9)
      CHARACTER*1    E(162000),OUT(162816)
      CHARACTER      R1*2,R2*3,R3*4,R4*5
      CHARACTER      R5*6,R6*7,R7*8,R8*9
      CHARACTER*8    H(102),R,BLANK
      INTEGER*2      ID(6,16,48,20),ND(45,90,20),IT2
      INTEGER*4      IT0,IT1,IT,IST,KT(4)
      EQUIVALENCE    (D(1,1,1,1),ID(1,1,1,1))
      EQUIVALENCE    (E(1),ND(1,1,1))
      EQUIVALENCE    (OUT(1),H(1)),(OUT(817),E(1))
      EQUIVALENCE    (G(1),R1),(G(1),R2),(G(1),R3)
      EQUIVALENCE    (G(1),R4),(G(1),R5),(G(1),R6)
      EQUIVALENCE    (G(1),R7),(G(1),R8),(G(2),R)
      EQUIVALENCE    (A(1),B(1)),(A(9),C(1))
      EQUIVALENCE    (C(21),TXS(1)),(C(25),TXM(1))
      EQUIVALENCE    (IT1,TS(1)),(IT2,TM(1))
*--------define constants
*        (0000 27 Nov 1989 = 628,128,000)
      IT0=628128000
      BLANK='        '
*--------display program identifier
      WRITE(*,800)
  800 FORMAT(////////,
     + 20X,'┌──────────────────────────────────────┐',/,
     + 20X,'│                                      │',/,
     + 20X,'│  ELSI DATA-FILE TRANSLATOR (TT)      │',/,
     + 20X,'│                                      │',/,
     + 20X,'│  Re-orders data into an array of     │',/,
     + 20X,'│  45 columns by 90 rows, discarding   │',/,
     + 20X,'│  overclocked pixels in rows 16 of    │',/,
     + 20X,'│  each section and columns 46-48.     │',/,
     + 20X,'│  Reads from input file ELSI.DAT      │',/,
     + 20X,'│  Writes to output file ELSI.IMG      │',/,
     + 20X,'│  Both files must already exist       │',/,
     + 20X,'│  in the default directory.           │',/,
     + 20X,'│                                      │',/,
     + 20X,'│  Press return to begin reading ...   │',/,
     + 20X,'│                                      │',/,
     + 20X,'└──────────────────────────────────────┘',/,
     + 3(/))
      READ(*,700)Z
  700 FORMAT(A1)
*--------open input and output files
      OPEN(50,FILE='ELSI.DAT',FORM='BINARY',
     + ACCESS='SEQUENTIAL',IOSTAT=IST,STATUS='OLD')
      OPEN(60,FILE='ELSI.IMG',FORM='BINARY',
     + ACCESS='SEQUENTIAL',IOSTAT=IST,STATUS='OLD')
*--------read and display file label
      READ(50)A
      WRITE(*,802)A
  802 FORMAT(
     + 20X,'Tape file label reads ...',/,
     + 8X,64(1H-),/,
     + 16(8X,64A1,/),
     + 8X,64(1H-),//,
     + 20X,'Press return to continue ...')
      READ(*,700)Z
*--------process 32 ELSI records (number in file)
*        Note: (5996544-1024 bytes)/185344 = 32.34
      WRITE(*,804)
  804 FORMAT(
     + 20X,'Pause before each record (Y/N)? [N] ',\)
      READ(*,700)Z
      WRITE(*,805)
  805 FORMAT(1X,/)
      IPAUSE=0
      IF(Z.EQ.'Y'.OR.Z.EQ.'y')IPAUSE=1
      DO 36 IREC=1,32
*---------read a record
      IF(IPAUSE.EQ.0)GOTO 2
      WRITE(*,806)IREC
 806  FORMAT(
     + 20X,'Ready to process record ',I2,':',/,
     + 20X,'Press return to continue ...')
      READ(*,700)
   2  WRITE(*,808)IREC
 808  FORMAT(
     + 20X,'Reading in record',I3,' ...',/)
      READ(90)B
      IF(IST.NE.0)GOTO 97
      READ(90)D
      IF(IST.NE.0)GOTO 97
      READ(90)C
      IF(IST.NE.0)GOTO 97
*---------decode time
      DO 3 I=1,4
      IF(I.LE.2)TM(I)=TXM(3-I)
   3  TS(I)=TXS(5-I)
      IT=IT)-110
      KT(1)=IT/3600
      IT=MOD(IT,3600)
      KT(2)=IT/60
      IT=MOD(IT,60)
      KT(3)=IT
      KT(4)=I/2
      IYR=1989
      IDM=27
*---------display time
      WRITE(*,810)IREC,KT
 810  FORMAT(
     + 20X,'ELSI record',I3,
     + 5X,'Universal Time: ',
     + I2,':',I2,':',I2,'.',I3,/)
*---------fill and display header
      DO 5 I=1,102
   5  H(I)=BLANK
      H(1)= 'ELSI    '
      H(2)= 'MJIR    '
      H(3)= 'AIN NO. '
      H(4)= '47434   '
      H(5)= 'TAPE /3 '
      H(6)= 'FILE /2 '
      H(7)= 'RECORD= '
      I=IREC+10000
      R=BLANK
      WRITE(R4,300)I
 300  FORMAT(I5)
      H(8)=R
      H(9)= 'INITIAL '
      H(10)='IMAGE IS'
      H(11)=' NUMBER '
      I=20*(IREC-1)+100031
      R=BLANK
      WRITE(R5,310)I
 310  FORMAT(I6)
      H(12)=R
      H(13)='DATE IS '
      H(14)='11/27/89'
      H(15)='  TIME: '
      R=BLANK
      I=KT(3)+100
      WRITE(R8,314)I
 314  FORMAT(6X,I3)
      I=KT(2)+100
      WRITE(R6,315)I
 315  FORMAT(3X,I3,1H:)
      I=KT(1)+100
      WRITE(R3,316)I
 316  FORMAT(I3,1H:)
      H(16)=R
      H(17)='  PLUS  '
      R=BLANK
      I=1000+KT(4)
      WRITE(R8,318)I
 318  FORMAT(I4,1X,'MSEC')
      H(18)=R
      H(19)=BLANK
      H(20)='DAVID J.'
      H(21)=' KNECHT '
      H(22)=' 3/26/90'
      WRITE(*,812)H
 812  FORMAT(
     + 20X,'New record header reads ...',/,
     + 16X,48(1H-),/,
     + 17(16X,6A8,/),/,
     + 16X,48(1H-),/)
```

```
*---------reorder 20 images
      WRITE(*,816)IREC
  816 FORMAT(
     * 20X,'Reordering record',I3,' ...',/)
      DO 20 IMAG=1,20
      DO 20 IROW=1,90
      II=IROW-1
      ISCT=II/15+1
      ILIN=MOD(II,15)+1
      DO 20 ICOL=1,45
      NO(ICOL,IROW,IMAG)=ID(ISCT,ILIN,ICOL,IMAG)
   20 CONTINUE
*---------write new record to file
      WRITE(*,818)IREC
  818 FORMAT(
     * 20X,'Writing out record',I3,' ...')
      WRITE(60)OUT
      WRITE(*,820)
  820 FORMAT(20X,40(1H-),/)
   30 CONTINUE
*---------all records processed
      WRITE(*,825)
  825 FORMAT(20X,'All records processed')
      GOTO 99
*---------end of file or read error
   97 IF(IST.GT.0)GOTO 98
      WRITE(*,830)
  830 FORMAT(20X,'EXIT: End of file')
      GOTO 99
   98 WRITE(*,832)
  832 FORMAT(20X,'EXIT: Diskfile read/write error')
*---------normal exit
   99 CONTINUE
      CLOSE(90)
      ENDFILE (60)
      CLOSE(60)
      WRITE(*,899)
  899 FORMAT(//,
     * 20X,'┌──────────────────────────────────────┐',/,
     * 20X,'│      END OF ELSI DATA TRANSLATOR     │',/,
     * 20X,'└──────────────────────────────────────┘',/)
      END
```

DS ROUTINE (DISTRIBUTIONS):

```
      PROGRAM DS
*       Find distribution of pixel values and jumps
*       in the ten combined (even-minus-odd) images
*       of an ELSI record.  The size of the distri-
*       bution bins is 8.
*       Version 0000 -  D. J. Knecht  09 April 1990
*
*---------variables
      CHARACTER*1 A(162816),D(162000),DS(16384)
      CHARACTER*1 REG(2),Z
      CHARACTER*72 SETH
      CHARACTER*8 H(102)
      INTEGER*2   NO(45,90,20),NOSP(4096),NOSN(4096)
      INTEGER*2   NV,NPOS,NNEG,IREG,MARK,IDUM
      EQUIVALENCE (A(1),H(1)),(A(817),D(1))
      EQUIVALENCE (NOSP(1),DS(1)),(NOSN(1),DS(8193))
      EQUIVALENCE (D(1),NO(1,1,1)),(IREG,REG(1))
*---------open files
      OPEN(60,FILE='ELSI.IMG',FORM='BINARY',
     + ACCESS='SEQUENTIAL',IOSTAT=IOC,STATUS='OLD')
*---------constants
      MARK=2573
*---------initialize
      INPUT=0
      ISUBT=0
      IDIST=0
      IREC=0
*---------display program identifier
    1 WRITE(*,800)
  800 FORMAT(/////////,
     + 20X,'┌──────────────────────────────────┐',/,
     + 20X,'│                                  │',/,
     + 20X,'│    DISTRIBUTION COMPUTER (DS)    │',/,
     + 20X,'│                                  │',/,
     + 20X,'│   Computes the distribution of   │',/,
     + 20X,'│   pixel values in an ELSI image  │',/,
     + 20X,'│   Reads input from file ELSI.IMG │',/,
     + 20X,'│   (must exist on default drive)  │',/,
     + 20X,'│   Writes to a file specified by  │',/,
     + 20X,'│   keyboard input.                │',/,
     + 20X,'│                                  │',/,
     + 20X,'|            SELECT OPERATION:                 |',/,
     + 20X,'|        R = Read an input record              |',/,
     + 20X,'|        S = Subtract backgrounds              |',/,
     + 20X,'|        V = Values distribution               |',/,
     + 20X,'|        J = Jumps distribution                |',/,
     + 20X,'|        O = Output to a file                  |',/,
     + 20X,'|        E = Exit                              |',/,
     + 20X,'|                                              |',/,
     + 20X,'|                                              |',/,
     + 20X,'└──────────────────────────────────────────────┘',/,
     + 20X,'            SELECTION:--> ',\)
*---------select operation
    2  READ(*,700)Z
  700  FORMAT(A1)
       IF(Z.EQ.'R'.OR.Z.EQ.'r')GOTO 4
       IF(Z.EQ.'S'.OR.Z.EQ.'s')GOTO 11
       IF(Z.EQ.'V'.OR.Z.EQ.'v')GOTO 41
       IF(Z.EQ.'J'.OR.Z.EQ.'j')GOTO 61
       IF(Z.EQ.'O'.OR.Z.EQ.'o')GOTO 81
       IF(Z.EQ.'E'.OR.Z.EQ.'e')GOTO 99
       GOTO 2
*---------read input record
    4  IF(INPUT.EQ.0)GOTO 5
       WRITE(*,804)
  804  FORMAT(
     + 20X,'Discard current record (Y/N)? [N] ',\)
       READ(*,700)Z
       IF(Z.NE.'Y'.AND.Z.NE.'y')GOTO 1
    5  IREC=IREC+1
       CALL RDREC(A,D,ND,IREC)
       INPUT=1
       ISUBT=0
       IDIST=0
       WRITE(*,808)(H(J),J=1,18)
  808  FORMAT(20X,'Record header:',/,
     + 20X,48(1H-),/,3(20X,6A8,/),20X,48(1H-),/,
     + 20X,'Press return to continue ...',)
       READ(*,700)Z
       GOTO 1
*---------subtract within each frame pair
   11  IF(INPUT.EQ.1)GOTO 12
       WRITE(*,812)
  812  FORMAT(20X,'No data have been read.',/,
     + 20X,'Press return for main menu ...')
       READ(*,700)Z
       GOTO 1
   12  CALL SUBTR(ND)
       ISUBT=1
       IDIST=0
       WRITE(*,814)
  814  FORMAT(
     + 20X,'Subtraction is completed.',/,
     + 20X,'Press return to continue ...',)
       READ(*,700)Z
       GOTO 1
*---------compute distribution of values
   41  IF(ISUBT.EQ.1)GOTO 42
       WRITE(*,842)
  842  FORMAT(
     + 20X,'Subtractions have not been done. ',/,
     + 20X,'Press return for main menu ...')
       READ(*,700)Z
       GOTO 1
   42  WRITE(*,848)
  848  FORMAT(/,
     + 20X,'Computing distribution for all pairs ')
       DO 43 I=1,4096
       NOSP(I)=0
   43  NOSN(I)=0
       DO 46 IPR=1,10
       K=IPR*2
       DO 46 J=1,60
       DO 46 I=1,45
       NV=ND(I,J,K)
       IF(NV.GE.0)GOTO 44
       NNEG=-NV/8+1
       NOSN(NNEG)=NOSN(NNEG)+1
       GOTO 46
   44  NPOS=NV/8+1
       NOSP(NPOS)=NOSP(NPOS)+1
   46  CONTINUE
       WRITE(*,850)
  850  FORMAT(
     + 20X,'Value distribution completed.',/,
     + 20X,'Press return to continue ...',)
       READ(*,700)Z
       IDIST=1
       GOTO 1
*---------compute distribution of jumps
   61  IF(ISUBT.EQ.1)GOTO 62
       WRITE(*,862)
  862  FORMAT(
     + 20X,'Subtractions have not been done. ',/,
     + 20X,'Press return for main menu ...')
```

```
      READ(*,700)Z
      GOTO 1
   62 WRITE(*,860)
  860 FORMAT(/,
     + 20X,'Computing distribution for all pairs')
      DO 63 I=1,4096
      NOSP(I)=0
   63 NOSN(I)=0
      DO 66 IPR=1,10
      K=IPR*2
      DO 66 J=1,60
      DO 66 I=2,45
      II=I-1
      NV=NO(I,J,K)-NO(II,J,K)
      IF(NV.GE.0)GOTO 64
      NNEG=-NV/8+1
      NOSN(NNEG)=NOSN(NNEG)+1
      GOTO 66
   64 NPOS=NV/8+1
      NOSP(NPOS)=NOSP(NPOS)+1
   66 CONTINUE
      WRITE(*,870)
  870 FORMAT(
     + 20X,'Distribution is completed.',/,
     + 20X,'Press return to continue ...',)
      READ(*,700)Z
      IDIST=1
      GOTO 1
*---------output to file
   81 IF(IDIST.EQ.1)GOTO 82
      WRITE(*,882)
  882 FORMAT(
     + 20X,'Distribution has not been computed.',/,
     + 20X,'Press return for main menu ...')
      GOTO 1
   82 WRITE(*,884)
  884 FORMAT(
     + 20X,'Printing distribution for ',/,
     + 20X,'all image pairs to a file.',/,
     + 20X,'Provide a file name ...')
      OPEN(77,FILE=' ',FORM='FORMATTED',
     + STATUS='NEW',ACCESS='SEQUENTIAL')
      WRITE(77,770)(NOSP(I),I=1,512),(NOSN(I),I=1,512)
  770 FORMAT(
     + ' TEN IMAGES:  BINS 000 - 512',52(1H-),/,
     +  2(16(16I5,/),1X,79(1H-),/),1X,79(1H-),//,
     + ' TEN IMAGES:  BINS 000 - 512',52(1H-),/,
     +  2(16(16I5,/),1X,79(1H-),/),1X,79(1H-),//)
      ENDFILE (77)
      CLOSE (77)
      GOTO 1
*---------normal exit
   99 CONTINUE
      WRITE(*,899)
  899 FORMAT(/,
     + 20X,'┌──────────────────────────────────────┐',/,
     + 20X,'│    END OF DISTRIBUTION COMPUTATION   │',/,
     + 20X,'└──────────────────────────────────────┘',/,
     + 4(/))
      CONTINUE
      END


      SUBROUTINE RDREC(A,D,NO,IREC)
      CHARACTER*1 A(162016),D(162000),DD,Z
      INTEGER*2 NO(45,90,20)
      WRITE(*,302)IREC
  302 FORMAT(20X,'Reading record',I3,' ...')
      READ(60)A
*     ----switch byte order
      DO 20 K=1,81000
      I=(K-1)*2+1
      J=I+1
      DD=D(J)
      D(J)=D(I)
      D(I)=DD
   20 CONTINUE
*     ----print data sample (col 1-15, row 1-60)
      WRITE(*,304)
  304 FORMAT(
     + 20X,'Print data sample to file (Y/N)? [N] ',\)
      READ(*,100)Z
  100 FORMAT(A1)
      IF(Z.NE.'Y'.AND.Z.NE.'y')GOTO 40
      OPEN(66,FILE=' ',FORM='FORMATTED',
     + STATUS='NEW')
      DO 30 K=1,2
      DO 29 J=1,60
      WRITE(66,700)(NO(I,J,K),I=1,15)
  700 FORMAT(15I5)
   29 CONTINUE
```

```
      WRITE(66,701)
701   FORMAT(75(1H-))
 30   CONTINUE
 40   RETURN
      END


      SUBROUTINE SUBTR(NO)
      INTEGER*2 NO(45,90,20),N1
      N1=-1
      DO 9 IPAIR=1,10
      K1=(IPAIR-1)*2+1
      K2=K1+1
      DO 9 J=1,90
      DO 9 I=1,45
      NO(I,J,K2)=NO(I,J,K2)-NO(I,J,K1)
      NO(I,J,K1)=N1*NO(I,J,K2)
  9   CONTINUE
      RETURN
      END
```

GG ROUTINE (IMAGE PROCESSOR):

```
      PROGRAM GG
*        Program GG-5  (Revision 5)
*        Reconstruction implemented by correcting the
*        combined image.  Corrects fumps of -6 through +6
*        times 128, with boundaries at 64 above and below
*        these values, based on distributions obtained with
*        Program OOO.  (D. J. Knecht   09 April 1990)
*
*--------Grey-scale plot routine.  Reads in an ELSI file
*        of 20 reordered images; combines them by
*        subtraction and summing; corrects the data for
*        stuck bits; processes the result to find and
*        adjust max, min, range, slope, threshold,
*        saturation; and writes the result to a grey-
*        scale print file for the laser printer.
*--------declare variables
      CHARACTER*1 A(162016),D(162000),Z
      CHARACTER*1 Q(92,100),P(47,90)
      CHARACTER*8 H(102)
      INTEGER*2   NO(45,90,20),MO(45,90),NJ,MARK,IOLM
      INTEGER*2   MIN,MAX,IRNG,ITHR,ISAT,ISLP,ITOP,IBOT
      INTEGER*4   NOUT(45,90,2),NO(2)
      EQUIVALENCE (A(1),H(1)),(A(817),D(1))
      EQUIVALENCE (P(1,1),MO(1,1))
      EQUIVALENCE (D(1),NO(1,1,1))
*--------open files
      OPEN(60,FILE='ELSI.IMG',FORM='BINARY',
     + ACCESS='SEQUENTIAL',IOSTAT=IOC,STATUS='OLD')
*--------define constants
      MARK=2573
      ITOP=159
      IBOT=32
*--------initialize
      INPUT=0
      ISUBT=0
      ISAVE=0
      IMEAN=0
      IPROC=0
      IREC=0
*--------display program identifier
    1 WRITE(*,800)
  800 FORMAT(////////,
     + 20X,'┌──────────────────────────────────┐',/,
     + 20X,'│                                  │',/,
     + 20X,'│   ELSI IMAGE PROCESSOR  (GG)     │',/,
     + 20X,'│        (Revision 5.0)            │',/,
     + 20X,'│  to produce grey-scale prints    │',/,
     + 20X,'│  Reads the input file ELSI.IMG   │',/,
     + 20X,'│  (must exist on default drive)   │',/,
     + 20X,'│  Writes output files named in    │',/,
     + 20X,'│  response to interactive query.  │',/,
     + 20X,'│                                  │',/,
     + 20X,'│       SELECT OPERATION:          │',/,
     + 20X,'│   R = Read an input record       │',/,
     + 20X,'│   S = Subtract backgrounds       │',/,
     + 20X,'│   C = Correct for stuck bits     │',/,
     + 20X,'│   A = Average selected pairs     │',/,
     + 20X,'│   P = Process composite image    │',/,
     + 20X,'│   O = Output to print file       │',/,
     + 20X,'│   E = Exit                       │',/,
     + 20X,'│                                  │',/,
     + 20X,'└──────────────────────────────────┘',/,
     + 20X,'        SELECTION:--> ',\)
```

```
*---------select operation
    2  READ(*,'(A1)')Z
       IF(Z.EQ.'R'.OR.Z.EQ.'r')GOTO 4
       IF(Z.EQ.'S'.OR.Z.EQ.'s')GOTO 11
       IF(Z.EQ.'C'.OR.Z.EQ.'c')GOTO 21
       IF(Z.EQ.'A'.OR.Z.EQ.'a')GOTO 41
       IF(Z.EQ.'P'.OR.Z.EQ.'p')GOTO 61
       IF(Z.EQ.'O'.OR.Z.EQ.'o')GOTO 81
       IF(Z.EQ.'E'.OR.Z.EQ.'e')GOTO 91
       GOTO 2
*---------read input record
    4  IF(ISAVE.EQ.1.OR.INPUT.EQ.0)GOTO 5
       WRITE(*,804)
  804  FORMAT(
     * 20X,'Discard current record (Y/N)? (N) ',\)
       READ(*,'(A1)')Z
       IF(Z.NE.'Y'.AND.Z.NE.'y')GOTO 1
    5  IREC=IREC+1
       CALL RDREC(A,D,ND,IREC)
       INPUT=1
       ISUBT=0
       IMEAN=0
       IPROC=0
       ISAVE=0
       WRITE(*,808)(H(J),J=1,18)
  808  FORMAT(20X,'Record header:',/,
     * 20X,48(1H-),/,3(20X,6A8,/),20X,48(1H-),/,
     * 20X,'Press return to continue ...',)
       READ(*,'(A1)')Z
       GOTO 1
*---------subtract within each frame pair
   11  IF(INPUT.EQ.1)GOTO 12
       WRITE(*,812)
  812  FORMAT(20X,'No data have been read.',/,
     * 20X,'Press return for main menu ...')
       READ(*,'(A1)')Z
       GOTO 1
   12  CALL SUBTR(ND)
       ISUBT=1
       WRITE(*,814)
  814  FORMAT(
     * 20X,'Subtraction is completed.',/,
     * 20X,'Press return to continue ...',)
       READ(*,'(A1)')Z
       GOTO 1
*---------correct for stuck bits (rows 1-60)
   21  IF(INPUT.EQ.1.AND.ISUBT.EQ.1)GOTO 22
       WRITE(*,822)
  822  FORMAT(20X,'No data have been prepared.',/,
     * 20X,'Press return for main menu ...')
       READ(*,'(A1)')Z
       GOTO 1
   22  WRITE(*,824)IREC,H(12)
  824  FORMAT(
     * 20X,'Correcting data of record',I3,/,
     * 20X,'The first image is number ',A8,/)
       DO 30 IPAIR=1,10
       K1=(IPAIR-1)*2+1
       K2=K1+1
       CALL RECON(ND,MD,K1)
       CALL RECON(ND,MD,K2)
   30  CONTINUE
       IMEAN=0
       IPROC=0
       WRITE(*,830)
  830  FORMAT(/,
     * 20X,'Correction is completed.',/,
     * 20X,'Press return to continue ...',)
       READ(*,'(A1)')Z
       GOTO 1
*---------average over successive pairs
   41  IF(INPUT.EQ.1.AND.ISUBT.EQ.1)GOTO 42
       WRITE(*,842)
  842  FORMAT(20X,'No data have been prepared.',/,
     * 20X,'Press return for main menu ...')
       READ(*,'(A1)')Z
       GOTO 1
*      ----specify summing
   42  WRITE(*,844)H(12)
  844  FORMAT(
     * 20X,'Image pairs are numbered 1-10.',/,
     * 20X,'The first image is number ',A8,/,
     * 20X,'Average starting with pair:  [1] ',\)
       READ(*,706)KBEG
  706  FORMAT(BN,I6)
       IF(KBEG.EQ.0)KBEG=1
       IF(KBEG.LT.1.OR.KBEG.GT.10)GOTO 42
       WRITE(*,846)
  846  FORMAT(
     * 20X,'      and ending with pair: [10] ',\)
```

```
      READ(*,706)KEND
      IF(KEND.EQ.0)KEND=10
      IF(KEND.LT.0.OR.KEND.GT.10)GOTO 42
      IF(KEND.LT.KBEG)GOTO 42
      WRITE(*,848)KBEG,KEND
 848  FORMAT(/,
     * 20X,'Combining pairs',I2,' - ',I2,' ...')
      CALL AVRAG(NO,NOUT,KBEG,KEND)
      IMEAN=1
      IPROC=0
      WRITE(*,850)
 850  FORMAT(
     * 20X,'Combining pairs is completed.',/,
     * 20X,'Press return to continue ...',)
      READ(*,'(A1)')Z
      GOTO 1
*---------process an averaged image
  61  IF(IMEAN.EQ.1)GOTO 62
      WRITE(*,862)
 862  FORMAT(20X,'Data have not been averaged.',/,
     * 20X,'Press return for main menu ...')
      READ(*,'(A1)')Z
      GOTO 1
  62  WRITE(*,864)
 864  FORMAT(
     * 20X,'Select result to print: ',/,
     * 20X,'   1 = Odd minus even frames',/,
     * 20X,'   2 = Even minus odd frames  [2] ',\)
      READ(*,706)M
      IF(M.EQ.0)M=2
      IF(M.LT.1.OR.M.GT.2)GOTO 62
      CALL PROCS(NOUT,M,P,Q)
      IPROC=1
      WRITE(*,866)
 866  FORMAT(
     * 20X,'Processing is complete.',/,
     * 20X,'Press return to continue ...',)
      READ(*,'(A1)')Z
      GOTO 1
*---------output results
  81  IF(IPROC.NE.0)GOTO 82
      WRITE(*,882)
 882  FORMAT(20X,'Data have not been processed.',/,
     * 20X,'Press return for main menu ...')
      READ(*,'(A1)')Z
      GOTO 1
  82  WRITE(*,884)
 884  FORMAT(
     * 20X,'Large or small image size (L/S)? [L] ',\)
      READ(*,'(A1)')Z
      ISIZE=2
      IF(Z.EQ.'S'.OR.Z.EQ.'s')ISIZE=1
      OPEN(62,FILE=' ',FORM='BINARY',
     * ACCESS='SEQUENTIAL',IOSTAT=IST,STATUS='NEW')
      IF(ISIZE.EQ.1)WRITE(62)P
      IF(ISIZE.EQ.2)WRITE(62)Q
      ENDFILE (62)
      CLOSE (62)
      ISAVE=1
      GOTO 1
*---------check status before exit
  91  IF(ISAVE.EQ.1)GOTO 99
      WRITE(*,892)
 892  FORMAT(
     * 20X,'Discard current work (Y/N)? [N] ',\)
      READ(*,'(A1)')Z
      IF(Z.EQ.'Y'.OR.Z.EQ.'y')GOTO 99
      GOTO 1
*---------normal exit
  99  CONTINUE
      CLOSE(60)
      WRITE(*,899)
 899  FORMAT(/,
     * 20X,'┌──────────────────────────────────────┐',/,
     * 20X,'│   END OF ELSI IMAGE PROCESSOR (GG)   │',/,
     * 20X,'└──────────────────────────────────────┘',/,
     * 4(/))
      END


      SUBROUTINE AVRAG(NO,NOUT,KBEG,KEND)
      INTEGER*2   NO(45,98,20),NV,MARK,ITOP,IBOT
      INTEGER*4   NOUT(45,98,2),NO(2)
*     ----clear registers and compute sums
      DO 4 M=1,2
      DO 4 J=1,98
      DO 4 I=1,45
   4  NOUT(I,J,M)=0
```

```
      DO 6 K=KBEG,KEND
      K1=(K-1)*2+1
      K2=K1+1
      DO 6 J=1,90
      DO 6 I=1,45
      NO(1)=NO(I,J,K1)
      NO(2)=NO(I,J,K2)
      NOUT(I,J,1)=NOUT(I,J,1)+NO(1)
      NOUT(I,J,2)=NOUT(I,J,2)+NO(2)
    6 CONTINUE
*     ----convert to averages
      NIM=KEND-KBEG+1
      DO 8 K=1,2
      DO 8 J=1,90
      DO 8 I=1,45
      NOUT(I,J,K)=NOUT(I,J,K)/NIM
    8 CONTINUE
      RETURN
      END


      SUBROUTINE IGO(N)
      CHARACTER*1 Z
      READ(*,'(A1)')Z
      N=-1
      IF(Z.EQ.'G'.OR.Z.EQ.'g')N=0
      RETURN
      END


      SUBROUTINE JTEST(IV1,IV2,LDEL,IDEL)
*---------symbols
*          IV1,IV2   data values (input)
*          IDP,IDN   IV2-IV1, IV1-IV2 differences
*          LDEL      last large jump 128*(+3,-3,+5,-5)
*          IDEL      size of this jump (output)
*          IT        threshold levels
*---------variable types
      CHARACTER*1  Z
      INTEGER*2    IV1,IV2,IDE,LDEL,IDEL,IT(14)
      INTEGER*2    ISIGN,IDUM,IJMP,MSG(2)
      DATA         IT/0,64,128,192,256,320,384,
     +             448,512,576,640,704,768,832/
*---------find size of jump
      IDE=IV2-IV1
      ISIGN=1
      IF(IDE.GE.0)GOTO 6
      ISIGN=-1
      IDE=-IDE
    6 IJMP=10
      IF(IDE.LT.IT(14).AND.IDE.GE.IT(12))IJMP=6
      IF(IDE.LT.IT(12).AND.IDE.GE.IT(10))IJMP=5
      IF(IDE.LT.IT(10).AND.IDE.GE.IT(8))IJMP=4
      IF(IDE.LT.IT(8).AND.IDE.GE.IT(6))IJMP=3
      IF(IDE.LT.IT(6).AND.IDE.GE.IT(4))IJMP=2
      IF(IDE.LT.IT(4).AND.IDE.GE.IT(2))IJMP=1
      IF(IDE.LT.IT(2).AND.IDE.GE.IT(1))IJMP=0
      IF(IJMP.GE.0.AND.IJMP.LE.1)GOTO 8
      IF(IJMP.GE.3.AND.IJMP.LE.5)GOTO 8
*       ----illegal or unusual jumps
      MSG(1)=ISIGN*IJMP
      MSG(2)=IDE
      CALL MESSG(1,MSG)
      IF(IJMP.EQ.10)IJMP=0
    8 IF(IJMP.LE.1.OR.LDEL.EQ.0)GOTO 10
      IDUM=(-1)*ISIGN*LDEL
      IF(IDUM.GT.0)GOTO 10
      MSG(1)=ISIGN*IJMP
      MSG(2)=LDEL
      CALL MESSG(2,MSG)
*       ----return the jump value
   10 IDEL=ISIGN*IJMP
      IF(IJMP.GT.1)LDEL=IDEL
   60 CONTINUE
      RETURN
      END


      SUBROUTINE MESSG(N,MSG)
      CHARACTER*1  Z
      INTEGER*2    MSG(2)
      DATA Z/'S'/
      IF(N.EQ.1)WRITE(*,301)MSG
      IF(N.EQ.2)WRITE(*,302)MSG
  301 FORMAT(25X,5(1H*),
     +'   Jump of ',I2,' (',I4,') encountered',\)
  302 FORMAT(25X,5(1H*),
     +'   Jump ',I2,' follows jump ',I2,\)
      IF(Z.EQ.'G'.OR.Z.EQ.'g')GOTO 9
      READ(*,'(A1)')Z
    9 RETURN
      END
      SUBROUTINE PROCS(NOUT,M,P,Q)
      CHARACTER*1 P(47,50),Q(92,100),Z,REG(2)
      INTEGER*2   NV,IREG,MARK,ITOP,IBOT
      INTEGER*2   MIN,MAX,IRNG,ITHR,ISAT,ISLP,IDUM
      INTEGER*4   NOUT(45,50,3)
      EQUIVALENCE (IREG,REG(1))
      ITOP=159
      IBOT=32
      MARK=[illegible]
      MIN=32767
      MAX=-32767
*     ----find parameters
      DO 4 J=1,50
      DO 4 I=1,45
      IF(NOUT(I,J,M).LT.MIN)MIN=NOUT(I,J,M)
      IF(NOUT(I,J,M).GT.MAX)MAX=NOUT(I,J,M)
    4 CONTINUE
      IRNG=MAX-MIN
      ITHR=MIN
      ISAT=MAX
    6 ISLP=ISAT-ITHR
      SLP=ISLP
      SLP=SLP/128.
*     ----display parameters
    8 WRITE(*,300)MAX,MIN,IRNG,SLP,ISAT,ITHR
  300 FORMAT(/,
     * 20X,'Image parameters:',/,
     * 20X,'          Maximum=',I6,/,
     * 20X,'          Minimum=',I6,/,
     * 20X,'            Range=',I6,/,
     * 20X,'            Slope=',F6.2,/,
     * 20X,'       Saturation=',I6,/,
     * 20X,'        Threshold=',I6,/)
*     ----adjust parameters
      WRITE(*,308)
  308 FORMAT(20X,'Adjust parameters (Y/N)? [N] ',\)
      READ(*,'(A1)')Z
      IF(Z.NE.'Y'.AND.Z.NE.'y')GOTO 12
      WRITE(*,310)
  310 FORMAT(20X,'New threshold:  ',\)
      READ(*,700)IDUM
  700 FORMAT(BN,I6)
      ITHR=IDUM
      WRITE(*,312)
  312 FORMAT(20X,'New saturation: ',\)
      READ(*,700)IDUM
      ISAT=IDUM
      IF(ISAT.GT.ITHR)GOTO 10
      WRITE(*,314)
  314 FORMAT(
     * 20X,'Error: saturation below threshold')
      GOTO 8
   10 ISLP=ISAT-ITHR
      SLP=ISLP
      SLP=SLP/128.
*     ----convert for printing
   12 DO 18 J=1,50
      DO 16 I=1,45
      NV=NOUT(I,J,M)
      IF(NV.LT.ITHR)NV=ITHR
      IF(NV.GT.ISAT)NV=ISAT
      NV=NV-ITHR
      V=NV
      V=V/SLP
      NV=V
      IREG=ITOP-NV
      IF(IREG.GT.ITOP)IREG=ITOP
      IF(IREG.LT.IBOT)IREG=IBOT
      P(I,J)=REG(1)
      I2=(I-1)*2
      J2=(J-1)*2
      DO 14 J1=1,2
      J3=J2+J1
      DO 14 I1=1,2
      I3=I2+I1
   14 Q(I3,J3)=REG(1)
   16 CONTINUE
*     ----add end-of-record marks
      IREG=MARK
      P(46,J)=REG(1)
      P(47,J)=REG(2)
      DO 17 J1=1,2
      J3=J2+J1
      Q(91,J3)=REG(1)
   17 Q(92,J3)=REG(2)
   18 CONTINUE
      RETURN
      END
      SUBROUTINE BOREC(A,D,NO,IREC)
      CHARACTER*1 A(162000),D(162000),DD,Z
      INTEGER*2 NO(45,90,20)
      WRITE(*,302)IREC
  302 FORMAT(20X,'Reading record',I3,' ...')
      READ(60)A
*     ----switch byte order
      DO 20 K=1,81000
      I=(K-1)*2+1
      J=I+1
      DD=D(J)
      D(J)=D(I)
      D(I)=DD
   20 CONTINUE
*     ----print data sample (col 1-15, row 1-60)
      WRITE(*,304)
  304 FORMAT(
     * 20X,'Print data sample to file (Y/N)? (N) ',\)
      READ(*,100)Z
  100 FORMAT(A1)
      IF(Z.NE.'Y'.AND.Z.NE.'y')GOTO 40
      OPEN(64,FILE=' ',FORM='FORMATTED',
     * STATUS='NEW')
      DO 30 K=1,2
      DO 29 J=1,60
      WRITE(64,700)(NO(I,J,K),I=1,15)
  700 FORMAT(15I5)
   29 CONTINUE
      WRITE(64,701)
  701 FORMAT(75(1H-))
   30 CONTINUE
   40 RETURN
      END


      SUBROUTINE RECON(NO,MO,K)
      CHARACTER*1 Z
      INTEGER*2   NO(45,90,20),MO(45,90)
      INTEGER*2   IV1,IV2,IDEL,LDEL,ITEM
      INTEGER*2   MAX,MIN,MAXX,MINN,N128,IZER
*---------constants
      MAXX=-32765
      MINN=32765
      IZER=0
      N128=128
*---------initialize
      DO 2 J=1,90
      DO 2 I=1,45
    2 MO(I,J)=IZER
*---------display image number
      WRITE(*,310)K
  310 FORMAT(/,20X,'Correcting image ',I2)
*---------reconstruct center column (60-row limit)
      LDEL=0
      IDEL=0
      DO 30 J=2,60
      I=23
      JJ=J-1
      IV1=NO(I,JJ,K)
      IV2=NO(I,J,K)
      CALL JTEST(IV1,IV2,LDEL,IDEL)
      MO(I,J)=MO(I,JJ)+IDEL
   30 CONTINUE
*---------fan right side
      DO 40 J=1,60
      NJ=0
      LDEL=0
      IDEL=0
      DO 38 I=24,45
      II=I
      III=II-1
      IV1=NO(III,J,K)
      IV2=NO(II,J,K)
      CALL JTEST(IV1,IV2,LDEL,IDEL)
      MO(II,J)=MO(III,J)+IDEL
   38 CONTINUE
   40 CONTINUE
*---------fan left side
      DO 50 J=1,60
      NJ=0
      LDEL=0
      IDEL=0
      DO 48 I=1,22
      II=23-I
      III=II+1
      IV1=NO(III,J,K)
      IV2=NO(II,J,K)
      CALL JTEST(IV1,IV2,LDEL,IDEL)
      MO(II,J)=MO(III,J)+IDEL
   48 CONTINUE
   50 CONTINUE
```

```
*---------use jumps to correct the data
      DO 52 J=1,60
      DO 52 I=1,45
      IDUM=MO(I,J)*N128
      MO(I,J)=NO(I,J,K)-IDUM
   52 CONTINUE
      WRITE(*,352)
  352 FORMAT(
     * 20X,'Data rows 1-60 have been corrected.')
*---------find max and min
      MAX=MAXX
      MIN=MINN
      DO 54 J=1,60
      DO 54 I=1,45
      IF(MO(I,J).GT.MAX)MAX=MO(I,J)
      IF(MO(I,J).LT.MIN)MIN=MO(I,J)
   54 CONTINUE
*---------offset the data
      WRITE(*,354)MAX,MIN
  354 FORMAT(
     * 20X,'Maximum and minimum values are ',2I5,/,
     * 20X,'Offset by a some amount (Y/N)? [N] ',\)
      READ(*,'(A1)')Z
      IF(Z.NE.'Y'.AND.Z.NE.'y')GOTO 60
   55 WRITE(*,356)
  356 FORMAT(
     * 20X,'Enter offset to be added to data): ',\)
      READ(*,766)IDUM
  766 FORMAT(BN,I6)
      IF(IDUM.LT.-1023.OR.IDUM.GT.1023)GOTO 55
      DO 56 J=1,60
      DO 56 I=1,45
      MO(I,J)=MO(I,J)-IDUM
   56 CONTINUE
      IDUM=(-1)*MIN
      MAX=MAX-MIN
      MIN=0
      WRITE(*,358)IDUM,MAX,MIN
  358 FORMAT(
     * 20X,'Values have been offset by ',I5,/,
     * 20X,'New maximum and minimum are ',2I5)
*---------replace old data with new
   60 CONTINUE
      DO 64 J=1,60
      DO 64 I=1,45
      NO(I,J,K)=MO(I,J)
   64 CONTINUE
      DO 68 J=61,96
      DO 68 I=1,45
      NO(I,J,K)=IZER
   68 CONTINUE
*---------exit
   99 CONTINUE
      RETURN
      END


      SUBROUTINE SUBTR(NO)
      INTEGER*2 NO(45,96,20),N1
      N1=-1
      DO 9 IPAIR=1,10
      K1=(IPAIR-1)*2+1
      K2=K1+1
      DO 9 J=1,96
      DO 9 I=1,45
      NO(I,J,K2)=NO(I,J,K2)-NO(I,J,K1)
      NO(I,J,K1)=N1*NO(I,J,K2)
    9 CONTINUE
      RETURN
      END
```