TELEMATICS and INFORMATICS Vol. 2, No. 4, pp. 321–330, 1985

0736-5853/85 $3.00 + .00

SYNTHETIC APERTURE RADAR PROCESSING ON A CRAY-1 S SUPERCOMPUTER

Malcolm L. Wolf
David J. Lewis
Douglas G. Corr

Abstract—A study has been performed to determine the feasibility of implementing a SAR processor on a CRAY-1 S Supercomputer. Algebraic expressions for the computational and data transfer loads have been developed. It is shown that the CRAY-1 S can support an efficient and fast frequency domain SAR processor.

A spaceborne synthetic aperture radar (SAR) acquires vast quantities of radar echo data which may be digitally processed to produce large area (100 km × 100 km), high-quality images of the Earth's surface with very high resolution, typically 30 m. Since the data acquisition may proceed day and night and in all types of weather, spaceborne SAR provides a uniquely powerful instrument in the modern science of remote sensing. However, because of the very high data rates and the need for extensive data processing, digital SAR processing places severe demands on both the computational power and the I/O capability of a modern computer system.

Systems Designers Scientific (SD Scientific) is the scientific part of Systems Designers plc, a leading UK based systems consultancy specializing in the most technically demanding areas of computing and information technology. SD Scientific has implemented several digital SAR processing systems for Space Department of the Royal Aircraft Establishment (RAE), Farnborough (U.K.) and data from SEASAT, SAR-580 and SIR-B have been processed. As a result of this experience, SD Scientific was commissioned by Space Department of the RAE to study the feasibility of implementing a SAR processor on the CRAY-1 S which was recently installed at the RAE. In a later section of this paper, a number of designs for such a processor are considered and image throughputs predicted. Also, a discussion of computational modelling is included; it is intended to be of general interest to anyone setting out to perform a computational sizing exercise on the CRAY-1 S or similar supercomputer. However, we begin below with a brief introduction to the problem of digital SAR processing.

DIGITAL SAR PROCESSING

The principles of SAR are introduced in a book by Hovanessian,[1] and SAR processing is discussed in a number of papers, e.g., Raney[2] and Barber.[3] The heart of digital SAR processing is matched filter correlation. In the raw SAR data, the echo from a particu-

Drs. Wolf, Lewis, and Corr are with the Consultancy Division at Systems Designers Scientific in Surrey, England. We would like to thank Dr. M. K. Dakin (SD Scientific) and Mr. B. C. Barber (RAE, Space Dept.) for many stimulating and helpful discussions.

lar target on the ground is inextricably mixed with that from many other targets. A digital SAR processor employs matched filter correlation to selectively discriminate, from the mass of data, just that signal from the target of interest. Targets may then be located to high resolution in ground coordinates.

Computational requirements for a high-throughput digital SAR processor are discussed by Jones et al.[4] and Jurkevich and Petty.[5] A typical current generation spaceborne SAR generates data at a rate of approximately 100 Mbit/sec., covering a 100 km × 100 km image area in approximately 15 seconds. Real-time precision processing of this data would require a computational rate of the order of thousands to tens of thousands of Mflops (1 Mflop = 1 million floating-point operations per sec.). Clearly, such real-time processing is beyond the reach of the current generation of general purpose supercomputers, and would require the development of special-purpose hardware. In the meantime, the new generation of spaceborne SAR systems is due to become operational in the late 1980's. These include the SAR onboard the Earth Resources Satellite, ERS-1, of the European Space Agency (ESA). As a result, there is a growing requirement for high-throughput digital SAR processors which can be implemented on currently available and field-proven technology.

The present paper summarizes some results of a detailed analysis of the feasibility of implementing a SAR processor on the CRAY-1S in particular for ERS-1 data.

COMPUTATIONAL SIZING ON THE CRAY-1 S

Accurate estimates of the CPU and I/O times required by an algorithm are central to the system design process. Such timing information provides the basis for an informed choice between different algorithms and hardware options. In addition, it defines the required data throughput rate and provides a necessary input to system optimisation.

Modelling CPU load on a vector processor

It is conventional to express the computational load imposed by an algorithm in terms of the number of additions, multiplications, etc. required. While this procedure is appropriate for a conventional scalar machine, it has less meaning on a vector machine like the CRAY-1 S. Indeed, it can be positively misleading, since the time taken to perform some "volume" of arithmetic on a vector machine depends upon whether it is performed in scalar or vector mode. In vector mode, the time involved also depends on the vector length. For example, on the CRAY-1 S simple arithmetic operations may run 10 times faster in vector mode than in scalar mode. Thus, on a vector processor, the vectorial structure of the computational load must be accounted for explicitly in order to develop useful loading models. The algorithm must be broken down into primitive vector operations, which are chosen to reflect the vector architecture of the machine, and classified according to vector length.

Timing model for vector operations on the CRAY-1 S

A good introduction to the characteristics of pipelined vector computers and the architecture of the CRAY-1 S can be found in Hockney and Jesshope.[6]

Vector registers in the CRAY-1 S are of length 64, so the processing of any vector operation on vectors of length n, greater than 64, requires a number of loops in which subvectors of length 64 are processed, plus a loop in which the remainder (if any) of the elements are processed. The number of loops, L, required for vector length n, is given by:

$$L = \mathrm{INT}[(n+63)/64]. \tag{1}$$

The CPU time required to process a vector of length n should comprise a term which is linear in the number of subloops, L, as well as a term which is linear in the number of vector elements to be processed, n. We will write the CPU time as:

$$T(n,L) = T_0 + LT_1 + nT_n. \tag{2}$$

T_0 is the overhead incurred in setting up the control structure for processing of the subvectors. T_1 is the constant overhead incurred within a loop. It comprises the times required to load the pipelines and issue the instructions within the loop, and it is independent of the number of vector elements which are to be processed. T_n is the time required per vector element to perform the vector operation once the control structure is set up and the pipelines are loaded. If we use the term "disjoint" to indicate pipelines which are not chained together, then the time T_n is simply 1 clock-period for every disjoint pipeline through which the vector elements must pass within the loop.

When n is large compared to the vector register length $T(n,L)$ can be approximated by

$$T(n,L) = n*t_{\mathrm{inf}} \tag{3}$$

where

$$t_{\mathrm{inf}} = T_n + T_1/64. \tag{4}$$

It is readily confirmed in any particular case that the CPU time for a vector operation as a function of length does indeed conform to the relationships of equations (1–4). The set-up time is found typically to be about 60 clock-periods (cp). In order to predict T_n and T_1 for a concrete example, it remains to model the scheduling of vector instructions within a subloop. Simple models of instruction scheduling, which take account of chaining and the hardware conflicts that can prevent it, generally give rise to very accurate predictions of CPU time on the CRAY-1 S.

Predictions of t_{inf} derived in this way for a number of simple vector arithmetic operations are shown in Table 1. Results are typically accurate to around 2% or better, and a similar accuracy is obtained for complex combinations of vector logical operations. Where anomalies do occur, as in the case of complex-complex multiplication, they are generally of the order of a few percent.

In general, therefore, accurate timings are accessible at the design stage without the prior need to code an algorithm.

PACKING AND UNPACKING OF DATA

Raw SAR data is collected by ERS-1 as complex samples in a format of 5 bits real and 5 bits imaginary. The CRAY-1 has 64-bit architecture, so the natural data format required for processing is as 2 64-bit CRAY words per complex sample. However, if the data were to be stored on disk in this format, it would result in the transfer between disk and central memory of large amounts of unwanted null information. Disc transfer rate for a DD-29 disk storage unit is 4 Mbyte/sec., that is $2.4*10^{-7}$ seconds/byte; with an allowance for head seek and settle times, the time to transfer a byte, t_{DAT}, is taken as $3*10^{-7}$ sec. Because the disk-transfer times are expected to be a very significant overhead, we consider the possibility of packing the range compressed data on disc into contiguous

Table 1. Summary of Predicted and Experimental Values of t_{inf} for Some Vector Operations on the CRAY-1 S.

t_{inf}	Theory	Experimental	Error (%)
Real-real multiply	3.50	3.43	2.0
Real-real addition	3.48	3.41	2.1
Real-complex multiply	5.73	5.63	1.8
Complex-complex addition	6.73	6.70	0.4
Complex-complex multiply	8.13	7.81	4.0
Third order polynomial	5.00	5.03	0.6
Fourth order polynomial	6.25	6.30	0.8
Fifth order polynomial	7.50	7.56	0.8

Definitive information on the timing of a piece of FORTRAN code running on the CRAY-1 S can be obtained using the CRAY system utility program CYCLES. From FORTRAN code as input, CYCLES generates comprehensive information on the translation route followed by the CRAY FORTRAN compiler CFT as well as the timing of the resulting machine code. Thus, CYCLES provides a very useful tool in the final stages of optimising sections of critical code.

sections of 16-bits, in order to economise on disk-to-memory I/O. This packing is assumed as the output from range compression will probably be 16-bit, although 8-bit may be acceptable.

- Unpacked form

  1 complex sample = 2*64 bit words = 16 bytes

  Therefore, the time to transfer 1 complex range compressed sample in unpacked form is

  $16 * t_{DAT} = 4.8 * 10^{-6}$ sec.
- Packed form

  1 complex sample = (16-bit + 16-bit) = 4 bytes

  Therefore, the time to transfer 1 complex range compressed sample in packed form is

  $4 * t_{DAT} = 1.2 * 10^{-6}$ sec.

Thus, by packing the data, the saving in transfer time per complex sample is $3.6 * 10^{-6}$ sec., ie. 288 clock periods. The CPU time required to unpack the data per complex sample is much less than 288 cp; it will, therefore, be advantageous to store data on disk in packed form. A simple coded FORTRAN unpack and integer-to-real conversion performed the unpack in 20 cp per complex element. It is likely that a carefully coded CRAY assembly language routine would improve on this performance.

If 8-bit range compressed data is acceptable further savings in space and I/O time are possible. Similar savings are made if the raw data is bit rather than byte packed.

RANGE COMPRESSION MODEL

Validation of the theoretical timing models was achieved by using a FORTRAN coded realistic model for the SAR range-compression algorithm which exploits the special vector processing features of the CRAY-1 S. The model reads byte-packed data from disc asynchronously using the CRAY FORTRAN extension BUFFER IN, unpacks and organizes the data in complex range pulses, performs range-compression, and then packs and buffers out the result. When used to compress 4000 pulses (approximately one fifth of the raw data required to produce an 80 km square ERS-1 image) the predicted CPU time (0.0243 sec. per pulse) agreed with the observed value to better than 0.5%.

This result demonstrates that there are no unforseen nonadditive effects tending to degrade performance. This fact encourages a high degree of confidence in the predicted CPU times for the other SAR algorithms, which contribute to the total processor times reported in the next section.

SAR PROCESSOR ON THE CRAY-1 S

The studies described here relate primarily to the SAR onboard ERS-1, but the results could regeneralize to other spaceborne Earth imaging side-looking SAR. The ESA-defined standard image products for the ERS-1 SAR are shown in Table 2.

Processor CPU time

Time domain algorithm. The azimuth compression can be performed either directly in the time domain or indirectly in the frequency domain. Time-domain processing is potentially more accurate but computationally more intensive.

The CPU timings predicted for a precision time-domain processor running on the CRAY-1 S with different replica update rates are collected in Table 3. These figures only contain timings for the major computationally intensive parts of an algorithm, and when auxiliary control calculations are included, a further overhead of about 5% may be expected.

The values in Table 3 may be contrasted with the predicted performance of Experimental SAR Processing Facility (ESPF). The ESPF, which was designed and implemented by Systems Designers Scientific for the Space Department of the RAE (Farnborough), is the time-domain processor described by Corr and Haskell[7] and Lewis, Barber and Corr.[8] It is implemented on a PRIME 750 with an attached FPS 120 B array processor and would take about 50 hours to produce an image equivalent to the multi-look product (ML) of Table 3. The processor is shortly to be transferred to a PRIME 9950 with an attached FPS 5320 array processor and, after tuning, is expected to run approximately 3 times faster on this hardware. It should be appreciated that while time-domain processing is relatively slow, it is highly accurate and produces imagery of a very high quality.

We note that updating the compression replica less frequently than every fifth pixel results in sacrifice of image quality for a sharply diminishing saving in computational time. It is seen from Table 3 that, even on the CRAY-1 S, a time-domain processor would require nearly 2 hours to produce the multi-look product. The fast-delivery product takes nearly 20 minutes of CPU time but will probably require longer than this since it is likely to be I/O bound. We have concluded that the CRAY-1 S is not suitable for the implementation of a high-throughput time-domain SAR processor for the ERS-1 SAR.

Table 2. ERS-1 Image Products.

	Image Specification		
Image property	Multi-look (ML)	Fast-delivery (FD)	Full-precision (FP)
Range resolution	30m	40m	30m
Azimuth resolution	30m	40m	8m
Pixel spacing	12.5m x 12.5m	20m x 20m	12.5m x 3.125m
Image size	80km x 80km	80km x 80km	80km x 80km
Number of looks	6	3 out of 8	1

For the fast delivery product the azimuth resolution will allow 8 independent looks to be generated. Only 3 of these are required to satisfy the radiometric resolution requirements.

Table 3. Predicted CPU for a Time-Domain SAR Processor on the CRAY-1 S.

Pixels generated between Replica Update	CPU time for image product		
	Multi-look (ML) (min)	Fast-delivery (FD) (min)	Full-precision (FP)
1	187	25	22.4 hrs
3	118	21	11.2 hrs
5	105	20	9.0 hrs
Disc in	30–39	11–14	19–30 min
Disc out	21	14	17 min

The two times for Disc in are, respectively, the minimum and the probable input times (a time-domain processor may have to read the range compressed data more than once).

Frequency domain algorithm. A frequency-domain processor utilizes the approximation that the azimuth compression replica can be treated as invariant over a sizeable number of pixels along an azimuth line, thereby enabling the compression to be performed by replica multiplication in the frequency domain. Provided the image quality specification is met, this approximation is acceptable.

We have designed a precision frequency-domain processor for the CRAY-1 S which incorporates two matched filters. It consists of a range-compression phase followed by an azimuth-compression phase. In the latter, the data required to produce the intermediate complex pixels is generated directly by interpolation from a buffer of transformed azimuth strips which is maintained in memory. The intermediate complex pixels are resampled to the required grid spacing in ground range and azimuth and, finally, converted to power pixels and multi-looked. The CPU times predicted for this processor are shown in Table 4.

The figures in Table 4 are calculated assuming that FFT's are performed using the optimized routine CFFT2 in version 1.13 of the $SCILIB library. Nearly 50% of the total CPU time in the frequency-domain processor is taken up by FFT's, and it is believed that new parallel FFT algorithms, currently under development by CRAY Research, may lead to further savings in this portion of the total time. The predicted timings show that fast frequency-domain SAR processing can be implemented on the CRAY-1 S. By way of comparison, we note that the frequency-domain SAR processor, implemented at the Jet Propulsion Laboratory, Pasadena (JPL) on a 32-bit mini with 3 attached AP-120 B's, should take of the order of three hours to produce an image comparable to the multi-look image considered here.[2] We now turn to the data-handling requirements for a frequency domain system.

Table 4. Predicted CPU Time for Frequency-Domain Processor on the CRAY-1 S.

	Image product		
	Multi-look (ML) (min)	Fast-delivery (FD) (min)	Full-precision (FP) (min)
CPU time	25	13	29
Disc in	16	10	14
Disc out	10	4	8

Memory and I/O capacity

The major data-handling problem encountered in frequency-domain SAR processing is the necessity to transpose ("corner-turn") the matrix of range-compressed (RC) data. For an 80 km × 80 km image the RC data is a total of about 350 Mbytes ordered as range pulses. However, the RC data are required in azimuth order for azimuth compression. We have considered two designs for the frequency domain process, a large memory system and a small memory system.

Large memory system. The simplest processor design we have considered (shown in Figure 1) avoids the necessity to "corner-turn" the RC data by holding it in RAM, rather than dumping to a linear storage medium prior to azimuth compression. Even on a maximal CRAY-1 S configuration with 8 Mwords of Buffer memory and a 32 Mword SSD, there is unlikely to be sufficient memory to hold the whole RC dataset of a large area. However, blocks of range pulses which cover the desired FFT length in azimuth can be processed independently. For example, for the multi-look image product, the desired FFT length is 1024 pulses.

Since a range pulse is about 4000 complex samples (16 bit real + 16 bit imaginary) the block of 1024 pulses of RC data is less than 16 Mbytes and can be packed into 2.0×10^6 words of 64 bit CRAY memory. The packing can be performed in about 1 sec. of CPU time. When all of the other memory requirements of the processor are accounted for the total requirement is less than 3 Mwords, and the design of Figure 1 could be comfortably accommodated on a CRAY-1 S configuration with 4 Mword of memory. A good configuration would be a 1 Mword CRAY-1 S with an I/O subsystem equipped with a 4 Mword buffer memory. Ideally the I/O subsystem should be configured with a block multiplexer I/O processor (XIOP) which would enable the raw data tapes and final image tapes to transfer data directly to the CRAY mainframe without going through the link to the front end. High-speed tape peripherals currently under development by CRAY Research should enable the high-density data tapes (HDDT) to be read directly under control of the XIOP, thereby avoiding the necessity of a prior tape transcription process. On such a system, the data management is extremely simple, and all I/O can be overlayed with processing. In this case, the total elapsed time for image production will be approximately equal to the CPU times of Table 4.

Small memory system. The second processor design we have considered (shown in Figure 2) includes an intermediate dump of the RC data to disc, necessitating a "corner-turn" operation. The method considered is to buffer a set of range-compressed pulses on output from range compression and then write them to disc as partial azimuth strips. On input to azimuth compression, sets of partial azimuth strips are read up from disc, and azimuth line segments are assembled of a length suitable for azimuth compression. These two operations are referred to as "half-turning," and, by means of double buffering and asynchronous I/O, they can be overlayed with processing if sufficient buffer space is available. With careful system optimization, this processor design can be accommodated on a basic CRAY-1 S/1000, fitted with the usual DCU-3 disc control units and DD-29 disc storage units, in such a way that it is theoretically CPU bound. In practice, because of the complexity of the data-handling, it is expected that some small overhead may be incurred for I/O. The processor requires concurrent access to three separate disk units as shown in Figure 3. The disk holding the raw SAR data receives raw data for image $(N+1)$ from the front end, while the azimuth compression phase of the processing of image N is in progress. Similarly, the disc holding the final image data transfers image

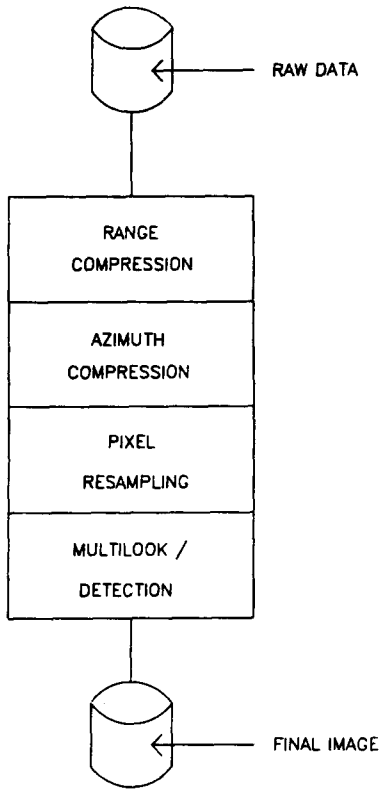

Figure 1. Schematic of design 1.

N-1 to the front end during the range-compression phase of the processing of image *N*. The processor requires a sustained raw data transfer rate from the front end of about 0.4 Mbyte/sec. in order to keep the processor busy. For the fast-delivery product, the transfer rate is nearly doubled.

CONCLUSIONS

We have shown that the timings of vector arithmetic and logical operations on the CRAY-1 S can be predicted to high accuracy using simple models based on a knowledge of the vector architecture of the machine.

We conclude that it is possible to implement an efficient frequency domain SAR processor on a CRAY-1 S. On a medium sized CRAY-1 S configured with I/O subsystem and XIOP, all data transfer could be overlayed with processing. Precision products could be produced in about 25 minutes with fast-delivery products in about 13 minutes. A processor design for the basic CRAY-1 S/1000 is slightly more complex because memory constraints necessitate an intermediate dump of range-compressed data to disc, however the processor should be capable of a similar image throughput provided the data transfer rate to the front-end can be sustained.

The throughput of precision SAR images predicted for the CRAY-1 S is encouraging. Clearly, a very impressive throughput is indicated on the CRAY X-MP or CRAY-2.

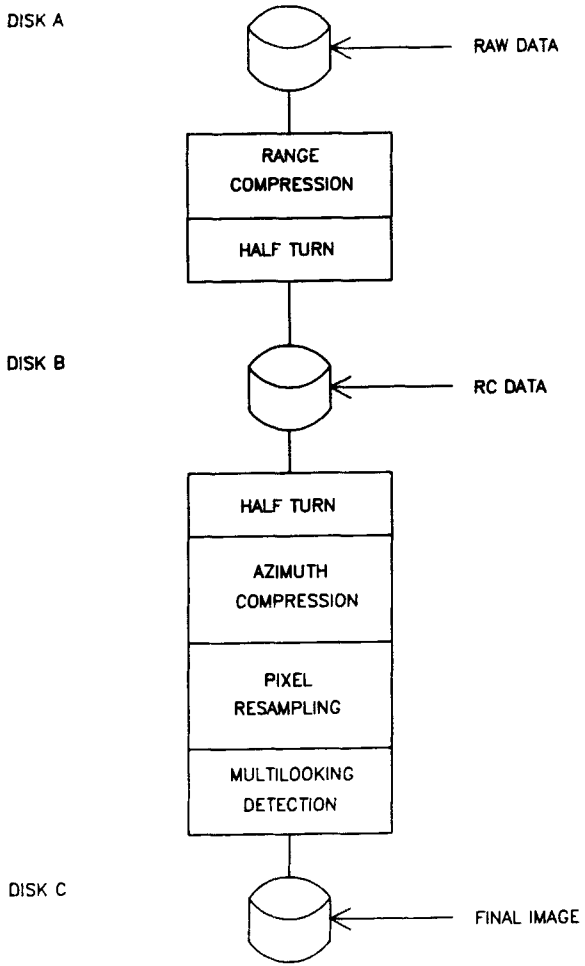

Figure 2. Schematic of design 2.

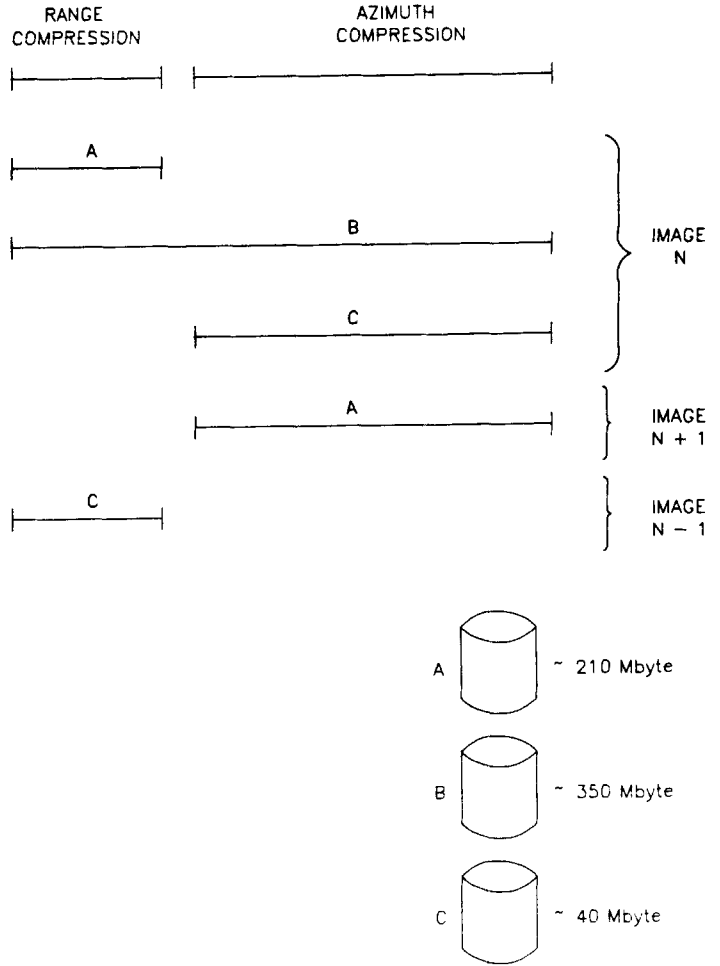

Figure 3. Disk requirements for design 2.

REFERENCES

1. Hovanessian, S. A. *Introduction to Synthetic Array and Imaging Radars*. Artech House Inc., Dedham, MA (1980).
2. Raney, R. K. *Int. J. Remote Sensing* 3, p. 243 (1982).
3. Barber, B. C. Royal Aircraft Establishment technical report 83079. *Int. J. Remote Sensing* 6, 7 p. 1009 (1983).
4. Jones, M. et al. *ESA Journal* 7, p. 145 (1983).
5. Jurkevich, I. and Petty, A. F. *Proceedings*, CRAY Image Processing Conference, 1983.
6. Hockney, R. W. and Jesshope, C. R. *Parallel Computers*. Adam Hilger Ltd. Bristol, England (1981).
7. Corr, D. G. and Haskell, A. *Proceedings, Int. Conf. on Spacecraft On-Board Data Management*, Nice (1978).
8. Lewis, D. J., Barber, B. C. and Corr, D. G. In *Satellite Remote Sensing*. Remote Sensing Society, Reading, England (1984).