Design of pipelined systems for landsat image processing

by DONALD C. BRABSTON and JOHN E. TABER
TRW Systems
Redondo Beach, California

INTRODUCTION

With two LANDSAT satellites in orbit and a third to be launched in the near future, the use of imagery obtained by satellite for monitoring the earth’s resources is well established. The timely, efficient, and routine handling of this image data is not so well established, however, and large systems for this purpose are just now coming into being. Two examples of such systems are NASA’s MDP system being developed by IBM, and Department of the Interior EROS Data Center’s EDIPS system being developed by TRW. Under consideration for the future is a system for handling imagery from LANDSAT D, whose data volume and throughput requirements are substantially more severe. This paper discusses some of the design considerations forced by the requirements of such systems and suggests solutions.

The main driving requirements for systems processing LANDSAT data are the extremely large data volume and the high throughput necessary to handle the data in a timely manner. These two requirements also separate the design of these systems from normal data handling systems which deal with lower volumes and throughput. The nature of the processing performed on LANDSAT data also differs from the processing performed in other applications. Special processing may include geometric warping of the imagery, radiometric correction, edge enhancement, and manual display interaction. Finally, the output products of these systems include annotated image films, and digital tapes of the imagery for specific users. The data volume and throughput, special processing, and special output products of LANDSAT data all press for unique solutions not found in other data processing systems.

Due to the high throughput and variety of special processing to be performed on the data, a pipelined design for the system is frequently the best choice. With a pipelined system, data from one scene may be read into the system while data from another scene is having special processing performed, and yet another scene is being output to film. Pipelining here results in maximizing system resource utilization and throughput while minimizing the total amount of hardware required. It is thus an efficient, high throughput, and cost effective design. Pipelining at a low level may mean the use of pipelined signal processors or microprocessor hardware. Pipelining at a high level involves the use of several general purpose computers, each of which is dedicated to performing a processing segment in the pipeline flow, in parallel with the other computers. What level of pipelining is best depends on the specific application and the hardware available. Figure 1 shows the processing flow of a typical pipelined system.

The next section discusses the specifics of LANDSAT image systems requirements; the third section briefly discusses special processing performed on the data; the fourth section discusses key design considerations and suggested solutions; and the last section summarizes the key points.

SPECIFIC LANDSAT IMAGE SYSTEM REQUIREMENTS

LANDSAT image processing system requirements are driven by the image sensors aboard the spacecraft. For LANDSAT's 1, 2, there are two sensors: The Multispectral Scanner (MSS) and a set of Return Beam Vidicons (RBV). The vidicons have not been operational since early in the mission. On LANDSAT C there is an MSS and a pair of higher resolution RBV's. The MSS has 26 detectors arrayed to image four spectral bands with a resolution of 79 meters and a fifth band with a resolution of 237 meters. In geometrically correcting the MSS scenes, the images are oversampled so that each picture element (pixel) is 57 meters square. An MSS scene is 185 kilometers square, so that an output image is 4/5 bands of about 3300^2 pixels. A similar computation yields a LANDSAT C RBV scene of about 5300^2 pixels, but only one band. A complete MSS scene is imaged roughly every 30 seconds, and the sensor is used over one hour per day. Combined data from LANDSAT's 1, 2, and C totals about 200 MSS scenes and 160 RBV scenes per day. Thus, a typical system requirement is to process this data in a two-shift working day. Total data volume is about 14 billion bytes per day. Allowing for gaps, spacecraft telemetry data, and annotation data (added in later processing) on the high density tapes on which the data is typically recorded, the system must be able to process data at a continuous rate of about 500,000 bytes per second.
In addition to throughput requirements, LANDSAT image processing has accuracy requirements on the special geometric and radiometric correction processing performed on the data. Typical requirements are that the output pixels be accurate to within one gray level (eight bit data having 256 gray levels) after radiometric and geometric correction, and that the geometric correction algorithm used introduce no more than \( \frac{1}{10} \) of a pixel spacing error (i.e., \( \frac{1}{10} \) of 57 meters for the earlier LANDSAT's) in the location of the output pixels. Further, in the case of geometrically correcting the data to register it to other LANDSAT images or to control points on the earth with known locations, the position error of output pixels may be required to be less than \( \frac{1}{2} \) or \( \frac{1}{5} \) pixel spacing. These accuracy requirements impose stringent conditions on the algorithms used, more than on the system design. However, the overall system design may well be impacted by the increased complexity and computation time of the sophisticated algorithms necessary to meet the accuracy requirements.

Reliability requirements for this type of system cannot be overlooked in the system design. System availability requirements of 85-90 percent are typical. Hardware redundancy, switchable peripherals, hot spares, and modular replacement of hardware components at the board level are all potential methods for meeting the reliability requirements and must be considered in the selection of a system design. Software reliability is also important. Detailed fail-safe error checking by the software modules, top-down design, and structured programming can and do help to insure reliable software, and are considered in the design and development of the system.

Although requirements on accuracy, reliability, special processing, etc., are important, the throughput requirement remains the most critical in selecting a system design. How these requirements determine the key issues in the system design will be discussed in the fourth section.

**SPECIAL PROCESSING**

The nature of LANDSAT imagery frequently requires special processing on the image data. This section briefly discusses some of the special processing, including geometric warping, radiometric correction, image enhancement, and format conversion of the input data. Although not all of these processes are required for every system, they are typical of the kinds of processing done on LANDSAT data and frequently drive aspects of the system design.

**Geometric correction**

The characteristics of the spacecraft, earth, and imaging sensor combine to distort the true ground data into the raw image data received from the spacecraft. This raw data must be corrected to show the true ground picture. The effects of earth rotation, spacecraft position and attitude, sensor scanning nonlinearities, earth curvature, and time delay between adjacent detector’s exposures must all be removed by the geometric correction processing. In addition, the output products are frequently required to be in one of several map projections, the most common being the Universal Transverse Mercator (UTM). Finally, the image data may be required to be registered to an earlier scene or to ground control points of known location. These effects are also allowed for in the geometric correction processing.

The geometric correction processing may proceed in two stages (which may be two segments of the pipeline). The first is the generation of a set of values at a grid of regularly spaced locations throughout the corrected image. The values specify the location of the corresponding pixels in the input, uncorrected image. The accuracy requirement drives the number of grid points, which in turn determines the computation and storage load for this portion of the processing. This grid is generated from the equations for the distortion effects mentioned above, the most significant being the spacecraft attitude and the map projection. The equations are complicated and their computation time must be carefully estimated.

Once the grid of locations is determined, the warping itself must be performed. Each pixel of the output image is located by interpolation from the grid of input locations in the input image to a fractional pixel location. The output pixel value is then interpolated from the values of its input pixel neighbors.

The complexity of the warping algorithm and its parameteric nature generally preclude its implementation purely in hardware, but the right choice of a hardware-software mix is not always obvious. The generation of the grid of location values in the first stage of the geometric correction involves computation of complex equations and is usually done in a general purpose computer. The nature of these computations, however, does impact the computer selection.

**Radiometric correction and image enhancement**

For the MSS sensor (and the TM sensor to be on LANDSAT D) there are multiple detectors with different characteristics. Therefore, different detectors will give different outputs for the same input radiances. Removing the detector...
specific errors in the pixel values is known as radiometric correction. This may be done by constructing a table for each detector which maps the pixel values received from the spacecraft into radiometrically correct values. Thus, for LANDSAT's 1, 2, and C there would be 26 tables of 64 elements each (since the data as received is 6 bits per pixel). As with geometric correction, this proceeds in two stages (which again, may be separate segments of the pipeline). The first stage is the construction of the detector-specific tables themselves, and the second is the mapping of the pixels through the tables. Construction of the tables may require obtaining statistics about the image data in the form of mean and variance or histograms of the pixel values or calibration data in order to remove image striping caused by the differences between detectors. In a general purpose computer, computation of these statistics requires several microseconds per pixel; hence, special purpose hardware or firmware is frequently dictated by the requirement to construct these tables.

Once the tables are constructed, the pixels must be mapped through the tables to generate the radiometrically corrected values. Since this again requires handling each pixel, special purpose hardware may be the solution to mapping the pixels at the required throughput.

Image enhancement may involve filtering the input (corrected or uncorrected) image to enhance edges, or remove haze or contrast stretching the image so that scenes of low contrast look better visually. These processes may take advantage of the scene statistics gathered at an earlier stage of the pipeline. Depending on the complexity of the algorithms used, special purpose hardware or firmware may be used, but again, normally a general purpose computer lacks sufficient speed to handle these operations. Figure 2 shows a corrected unenhanced image compared to the same image after edge enhancement.

Format conversion

LANDSAT image data as received from the satellite is interleaved by detector in a format called ‘band-interleaved-by-pixel’ (BIP), i.e., one pixel is received from each detector before the second pixel is received from each detector. Since an image line is formed by one detector as it scans the earth, in order to reform the image into image lines, the data format must be converted first into lines (band-interleaved-by-line or BIL) and finally into band-sequential (BSQ) where all lines of an MSS (or TM) spectral band are consecutive. Converting from BIP to BIL again involves handling each pixel, but only moving it from one location to another. Thus, specially addressable memories or computer controlled direct memory access devices (DMA’s) may be the solution here. For 26 detectors of 2300 pixels per input line, a memory of 60,000 bytes would be required—quite reasonable for modern minicomputers. Conversion from BIL to BSQ, however, could require a memory large enough to hold an entire scene (31 million bytes for the earlier LANDSAT’s) unless enough devices were available in the next pipeline segment to handle all bands simultaneously. Thus, a mass storage device such as a disk is usually needed.
here. Formatting this storage device so that image data is written on in BIL and later read off in BSQ, both at the required data rates, is a nontrivial task. Disk formatting will be discussed in more detail in the next section.

The special processing described in this section often necessitates special purpose hardware or firmware for handling the individual pixels. Even where pixels are not handled directly, the algorithms involved may impact hardware selection. The next section discusses further these and other key design issues.

KEY DESIGN ISSUES

The requirements of throughput, accuracy, reliability, economics, and special processing described above, force the selection of a system design. This is not to say that there is only one solution to these problems, but that there are certain key design issues which must be dealt with. Some of these, such as the choice of special hardware and memory sizing have been mentioned above. Other issues are the selection of general purpose computers, the need for a process control scheduler to handle the pipeline flow, efficient use of mass storage devices, and software development methodology. Obviously all of these cannot be discussed in depth here due to lack of space; however, this section looks briefly at each issue, showing the problem, and suggesting a method or methods for its solution.

System loading considerations

The throughput requirement of hundreds of kilobytes per second frequently means many transfers per second through a general purpose computer's memory, perhaps several times the overall throughput rate. For example, suppose the system of Figure 1 were to be implemented on one general purpose computer. The transfers shown in Figure 3 may then be executed for each scene. Although the pipeline segments' processing occurs sequentially for a given scene, normally the pipeline will be kept full and all of these segments will be executing on different scenes simultaneously.

In this example, ten transfers to and from memory are executed per pixel. At a data rate of, say 700,000 bytes per second on the average, this means an average data rate of 7 million bytes per second. In practice, the actual peak data rate may be substantially higher than this. For example, although the high density tape (HDT), film generator, and array processor transfer at a fairly constant rate, the disk and tape transfers occur much faster while actually performing the record transfers than the average rate which includes inter-record gaps. Thus, the transfers 2, 3, 7, 8, and 10 may reach 1.2 million bytes per second each. This gives a total peak rate of 9.5 million bytes per second (or actually more since the CPU normally fetches instructions and data from this memory as well). Thus, system loading considerations may force the division of the flow into specific pipeline segments independent of the allocation of processing functions to segments. This requires a memory with an access time of about 100 nanoseconds per byte. Furthermore, in the case of many minicomputers (e.g., PDP-11, ECLIPSE, SEL 32, etc.), the data must travel via one memory bus; hence, this bus must be capable of sustaining the data rates calculated here.

One potential solution to this problem is the selection of a computer with several data busses or even high speed I/O channels dedicated to transfers between certain peripherals and memory. Still another solution might be the selection of several banks of memory, each with its own memory controller to give independent parallel access to each bank. With this solution, (assuming the memory bus or busses could handle the data rate) buffers could be placed in different memory banks to reduce the load on each individual bank to an acceptable level. A variation on this solution would be to interleave the memory banks with a least-significant-bit addressing scheme to reduce the effective access time of the memory. This has a particular advantage here since most transfers are of image lines which transfer from successive locations in memory. A final solution might be to construct the process control software described later to preclude processes which strain the system's memory or bus capacity from running in parallel. If all of these potential solutions fail the analysis, more computers may be used, even to the ultimate extent of having one computer per pipeline segment. (Unfortunately, this may dictate still another computer to provide process control for the pipeline flow.)

A further problem occurs in considering the computational load on the computer's central processing unit (CPU). This is related directly to the software used by the system.

Current minicomputer operating systems, for example, have software overhead times of ½ millisecond per interrupt. If the system of Figure 3 receives an interrupt on completion of the transfer of every image line, there will be an interrupt about every 4 milliseconds for each of the ten transfers. This allows .4 msec for processing each interrupt, which may well be beyond the capacity of the system. Potential solutions to this are to get a new, faster operating system or to write device handlers which speed the interrupt processing or even trigger the start of the next transfer when the current one is complete. Another measure would be to generate an interrupt only after the transfer of several lines. Whether this is possible depends on the image data formats and the peripheral and computer hardware.
Hardware/software/firmware tradeoffs

The variety of special processing being performed on image data requires a mixture of general purpose computers and their software, firmware in the form of microprocessors or array processors, and special purpose hardware. In nearly all large scale, pipelined systems there exist a need for a process controller. Due to the complexity of this scheduler and the need to control the various peripherals, a general purpose computer will normally form the heart of the system. This computer may also be well-suited for interacting with operators and users via displays or terminals. In addition, it may perform some complex, low throughput special processing such as the generation of the grid of geometric correction locations or the radiometric correction tables from the scene statistics described earlier.

However, a general purpose computer is not normally suited for processing that deals with pixels on an individual basis, due to a lack of the speed required by such processing. Thus, either special firmware of hardware must be employed. The choice of firmware or hardware depends primarily on the complexity of the processing and the throughput required. As an example, the more complex functions of geometric warping and edge enhancement would probably be done in firmware if the required throughput could be met, while the format conversion and contrast stretching could be done in hardware. The gathering of scene statistics and radiometric correction could be done in either hardware or firmware, depending on cost effectiveness. Very complex algorithms such as geometric correction could be done in commercially available array processors (such as Floating Point Systems AP-120B, Signal Processing Systems SPS-81, or CSP, Inc.’s MAP-300). If necessary to meet the throughput requirements, an image can be split into parallel streams, each passing through a separate array processor with the output being merged again (if the algorithm allows such a split). These array processors are sufficiently complex that configuring them with the proper memory and most suitable options raises design issues on its own.

The decision to use special hardware or firmware in various parts of the system may also force segmentation of the pipeline at specific points. This and cost-effectiveness considerations make this a key design issue.

Process control

One disadvantage of a pipeline system design is the need for a more complex process control program or scheduler. On a strictly sequential system resource conflicts do not usually occur since only one process is in progress at any time. Similarly, a totally parallel system in which all resources are duplicated does not normally give resource conflicts. However, in even a simple pipelined system such as that of Figure 1, resource conflicts do occur, and the scheduling of the pipeline processing segments must be done. In Figure 1, for example, several resource conflicts are apparent: Scene 2 cannot begin its image corrections immediately after input because the array processor required for it is in use for Scene 1; similarly, conflicts arise in the case of the output film generation hardware. Other conflicts may arise not in the use of specific hardware, but in the total system load. If several special processing segments had the hardware to execute simultaneously, they might still exceed the load capacity of the system as far as memory or bus loads are concerned. Further, the scheduler may need to prevent other processes from executing concurrently because of conflicts in accessing shared data areas. In short, in a pipelined system a scheduler may be necessary to eliminate resource conflicts. (If the processing flow is static and the same for all scenes, the scheduler may be very simple or even nonexistent in that the individual pipeline segments tell each other when they are complete or available; but for more complex systems, the scheduler may become quite involved.)

The main functions of a scheduler in this type of system are: to determine the processing flow for each scene, initiate processes, determine processing conflicts and resolve them, and to monitor the processes for termination. All processes can communicate with the scheduler to indicate termination or error conditions, and in general need not communicate directly with other processes. Other scheduler functions may include operator interaction to determine each scene’s processing flow and to report error conditions. If the system runs out of resources, the scheduler may stop the flow altogether as the pipeline “backs up.” For example, in Figure 1, if the film output generator fails, later scenes will be unable to complete their processing which means that they must be stored on mass storage devices. As these devices fill up, the image correction, special processing, and input segments have no place to store their data, so that finally input must stop. If the input is from tape, data may not be lost; however, if the input is directly from the satellite, data may be lost at the input end of the system. (The system designers could, of course, construct the scheduler so that data is not output to film, but is saved on tape, or other measures to prevent the loss of data.)

The internal design of a scheduler is beyond the scope of this paper, but a scheduler is one solution to the problem of process control, which is an important design consideration.

Use of mass storage devices

In order to keep the process running smoothly, it is frequently necessary to use mass storage devices such as disks as buffers between pipeline segments. Otherwise, lack of a needed resource for even a brief period may bring all processing to a halt as main memory buffers fill up rapidly. In addition, disks may be necessary to store image data written in one order to be read in another order. Since industry standard 80- and 300-million byte moving-head disks are commercially available, these are good choices for mass storage buffers. Several issues arise in using these disks efficiently.

The data formatting on the disk is a design tradeoff which must be made. Figure 4 shows a typical disk format for buffering MSS data. Each track of the disk is hardwired as
WRITE 19, SKIP 2, 5 TIMES PER CYLINDER

MSS 4 LINES PER TRACK, 20 LINES PER CYLINDER

23 sectors of 768 bytes each. There are 5 tracks per cylinder. There is no time penalty for disk accesses which change tracks within a cylinder, but there is a substantial penalty for switching cylinders which requires a movement of the read/write head. If each image line is 3500 bytes, then 4 lines will fit into 19 sectors. Since the disk rotates at 3600 rpm, it takes about 14 msec to read 4 lines of data. Then 2 sectors are skipped to allow about 1½ msec for software overhead before the next 4 lines are read. Twelve sectors are left empty at the end of the cylinder to allow time for the head movement to switch to the next cylinder. Thus, the reading of 20 image lines requires on the average of 5 disk revolutions. This gives a sustained throughput capacity of 840,000 bytes per second, which is in excess of the required 700,000. Hence, there is some spare time available for situations in which system loading prevents being able to read the next 4 lines within 1½ msec after the last 4 are completed.

Because of the high speed involved, there is no time available for retries of disk reads or writes in the event of parity errors. However, with modern disks, the parity error rate is so low that the expected number of errors affects the image significantly. Thus, parity checking need not be performed. If parity checking is performed, retries should not be made, and a system alarm should be generated only if the error rate becomes too high. Parity checking, formatting, high speed software device handlers, and reliability are some of the key design issues in using mass storage devices efficiently.

Software methodology

In designing high throughput, reliable systems, the hardware design is not the only consideration. The software design and implementation for general purpose computers and for microprogrammable microprocessors and array processors is also extremely important. Software design problems arise in attaining the high throughput required, and in developing reliable software which is free of timing-dependent bugs. Further, as research progresses, new algorithms are developed, and new user needs become apparent, the requirements for LANDSAT image processing change. Therefore, the software should be modular enough to permit easy modification or inclusion of new processing. The scheduler should be sufficiently flexible that addition of new segments in the pipeline does not cause major redesign or recoding of the scheduler.

One means of attaining the high throughput required in array processors is to actually code and benchmark the critical high speed inner loops of the algorithms in the design phase, rather than waiting until the coding phase. This way, a good estimate of the actual throughput of these processing segments can be obtained early.

In developing reliable software, the top-down design process is very valuable. Software tools which aid this process are very beneficial in easily developing a top-level design and expanding it to deeper and deeper levels until the software can be coded directly from the design document. If done properly, the detailed design document also becomes the main software documentation. Two good tools for software design are structured flow charts and Program Design Language (PDL). Impressive cost savings in software development through the use of PDL have been cited. Software design tools aimed at aiding pipelined systems design would be of great benefit. These tools would facilitate checking for deadlocks, use of common data areas, etc. Unfortunately, we know of no such tools available at the present time.

Structured programming is also of benefit in developing software for pipelined systems, and especially for modifying it in the event of changing requirements. In using FORTRAN for the software development, the use of a structured FORTRAN preprocessor can be of benefit, especially when the design was done in a top-down structured fashion.
SUMMARY

This paper has shown how the requirements of large scale LANDSAT image processing systems force the consideration of certain key design issues. The main requirements of the system are the large data volume and high throughput, reliability, need for special processing, and accuracy. Processing peculiar to the LANDSAT systems includes geometric and radiometric correction, image enhancement, and high speed data reformatting. A pipelined design is often a natural choice for these systems. Although there are many issues to be tackled in the design of such systems, only the most important ones have been addressed here. These include system loading considerations in selecting computer hardware; the decision on where to use special hardware or array processors; the need for process control to schedule the pipeline's segments; the efficient use of mass storage devices; and software tools and techniques.

REFERENCES
