MVA '96 IAPR Workshop on Machine Vision Applications, November. 12-14, 1996, Tokyo, Japan

A Coarse Parallel Image Computing System for Remote Sensing

Joji Iisaka
Canada Centre For Remote Sensing
Department of Natural Resources Canada

Abstract

This paper discusses with the feasibility and value of a parallel image computing system for analyzing remotely sensed data on a Personal Computer Local Area Network. The design and configuration of the system are presented. The scheduling methods are also described. Examples of remote sensing analysis procedures that benefit from this type of system are illustrated.
It is concluded that the parallel image computing system is economical, viable, and well suited to remote sensing, but needs to be optimized for the system and to utilize more complex processing approaches to take full advantage of parallel computing.

1 Introduction

Remote sensing is of great importance around the world, especially for the purposes of earth resource and environment management. The analysis of remote sensing data is complicated involving many operations that require operator input and that are time consuming. In the past decades there has been an increase in the number of data sources available along with better resolution in spectral, spatial and time domains, contributing to the availability of an increasingly larger volume of data. Parallel image computing is expected to provide a means to extract this information effectively and to increase the speed and complexity of information used in the analysis of remotely sensed data.

This paper discusses the implementation of a pilot system based on a Personal Computer (PC) Local Area Network (LAN).

* Address: 588 Booth St., Ottawa, Ont., Canada
E-Mail: Joji.Iisaka@geocan.nrcan.gc.ca

2 Conventional Image Analysis vs. Parallel Image Computing for Remote Sensing

Conventional image analysis methods for remote sensing, such as multi-spectral analysis, have been developed using mostly pixel-by-pixel based methods, and little spatial information processing functions have been integrated into the systems. Most importantly, conventional methods are sequential (Ref.2,6,7). Furthermore, these procedures still require a significant human operator interaction and they have far to go to reach the goal of automated feature extraction from remotely sensing data.

A parallel-based system for computing image data is expected not only to increase speed (Ref. 1), which allows data to be processed faster (thus, saving valuable time), but also to achieve more complex feature extraction by employing data and information fusion processes in parallel.

With the increase in the amount of data available, it becomes necessary to process swiftly those data so that it may used more efficiently. Much of the feature extraction processing of image data is parallel in nature, which yields a natural advantage over an isolated single computer. It is necessary to overlay or *fuse* different types of information or data to extract terrain features of higher level categories (e.g. river or roads) rather than to extract lower level image features (e.g. bright pixels or lines) (Ref. 3,4,5). High-end computers with a single processor may not be particularly well suited to the specific type of computation required. Multiple lower-end computers operating in parallel offer a much more realistic and practical solution. A small parallel computer system need only consist of several low-end computers maintained in a network configuration.

A single computer processing data would have the output of one image channel sitting idly while it processed the next input channel. A parallel computer system could have separate computers processing separate image channels simultaneously resulting in little or even zero idle time. Furthermore, networks have been popping up almost everywhere. It is now possible to exploit the idle cycles of some machines resulting in virtually free processing time.

3 Design and Configuration

The initial parallel image computing design calls for a Local Area Network (LAN) consisting of three to five PCs. The computers would run many in-house developed image processing programs as well as an off-the-shelf program called WiT that comes with a built-in parallel execution ability (Ref. 8,9). The system runs under an object-oriented visual programming environment for designing computer algorithms with executable block diagrams that use icons and links that symbolize functions and data flow, respectively.

Five computers are currently being used: Each of these computers has varying processors, amounts of Random Access Memory (RAM), and operating systems. They are connected to each other by an Ethernet LAN (See Figure 1).

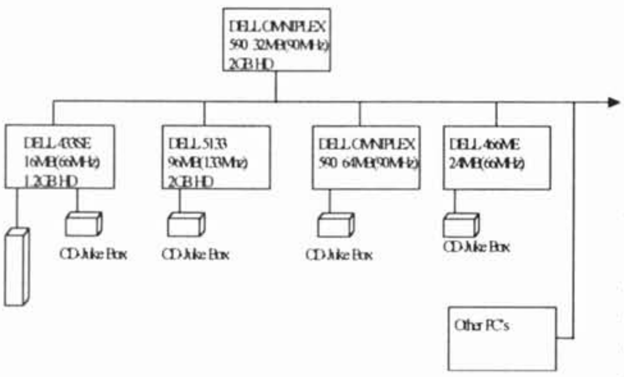
DELL OMNIPLEX 590 32MB(90MHz) 2GB HD
DELL 433SE 16MB(66MHz) 1.2GB HD
DELL 5133 96MB(133Mhz) 2GB HD
DELL OMNIPLEX 590 64MB(90MHz)
DELL 466ME 24MB(66MHz)
CD Juke Box
CD Juke Box
CD Juke Box
CD Juke Box
Other PC's

Figure 1: Layout of PCs in LAN for coarse parallel image processing.

The supervisor program protocol dictates that one computer run the Graphical User Interface (GUI) and the others run the Remote Access Servers (RAS). A PC named as PCIMAGE will run the GUI and two to four of the other machines will run the RAS. When a program drawn graphically as a data flow diagram is executed, the scheduler determines which machines will execute which portions of the program.

Each processor operates independently of the others. The current setup also consists of parallel computing on heterogeneous clusters as opposed to homogeneous clusters. The former describes the situation where all processors are the same and the latter where different processors are combined to solve a single problem. A heterogeneous environment is generally slower since conversion of data and messages is necessary between computers. The current network environment is a bus network. All the processors are connected on a single Ethernet line. To communicate with another, a computer broadcasts its message on the bus. Each computer has collision detection so that when two computers try to send messages at the same time, their messages are not lost. The design is very effective so long as the utilization of the network remains relatively low by other non-related functions.

4 Scheduling Methods

The scheduler decides which operators to run on which servers to maximize parallelism. There are two types of scheduling modes, flat and hierarchical, further enhancing the flexibility of parallel computing.

The scheduler is able to analyze link connections and determine how to dispatch operations effectively across the network of servers. An operation is executed by sending the required data objects to another server along with instructions on what to do with those data. While that server is busy, the scheduler dispatches more data and another instruction to another server. When a server is finished and comes back with its results, it is dispatched another instruction. The scheduler knows what servers are capable of executing what operations.

Data are generally not transported between the GUI and the servers unless it is necessary. Sometimes an entire algorithm, consisting of input, operations, and outputs, can be executed by a single server without any data transfer at all. Data transfer is carried out when an image or other piece of data needs to be displayed and when a server that has data cannot perform the operation required of it. The scheduler maintains the location of where each data object resides. When an operator needs to be scheduled, it is assigned to the available server that requires the least amount of data transfer. If a link divides the network into multiple parallel branches, the data will be physically copied to multiple servers. After the copying, the servers will execute in parallel.

Operators execute in the order they become ready, unless they become ready at the same time at which point the scheduler is free to execute any one first. User interface operators are always scheduled to execute whenever any of its inputs are ready. Another operator is ready when all of its inputs are available. After execution it sends its output to all of its descendants. If a descendant is still processing some previous inputs then the ancestor waits to send its output to the descendant until it has completed its execution.

There are two modes of scheduling, flat and hierarchical. The latter only applies to a program with hierarchical operators. They are used to facilitate the repetition of commonly used portions of a program. In flat scheduling mode, the scheduler is free to choose any available operators to execute regardless of which program the operators belong to. Data enter a hierarchical operator as soon as it is available and are executed immediately. Outputs are sent away as soon as they are ready. In hierarchical scheduling mode, the operators behave as though they were primitive operators.

The main program handles its data conversion and communication transparently using IPC (Inter-Processor Communication). Actual data transfer between the GUI and the servers is done only when absolutely necessary. The standard used for data distribution is eXternal Data Representation (XDR) produced by the Open Look Alliance. It specifies standards for encoding data so that they may be exchanged between any computer architectures.

5 Examples

There are many examples that demonstrate the ability of parallel image computing. Some of these are multi-spectral channel processing and spatial feature extraction through pixel swapping. Each of these examples is parallel in nature and can thus benefit from parallel computing (Ref. 3,4,5).

Multi-spectral channel processing involves processing the separate image channels of a data set to extract information. Essentially, it entails performing the same or similar operations on each of the channels of an image. This example clearly demonstrates one of the most simple cases of parallelism. Parallelism is advantageous here because the six completely independent operations can each be performed on a separate computer. Parallel computing also allows a channel to be used immediately for other tasks after being processed rather than waiting for all six to be done.

Pixel swapping is a very complicated example of the benefits of parallel computing. Pixel swapping extracts spatial entities like points, lines, regions, inner regions, and boundaries from a binary image (Ref. 3,4,5). As seen in Figure 2, data from some operators are often used by many other operators. Using a parallel computing system, any output that is used by more than one operator can be computed in parallel. For example, unaryOp #1 sends its output to display #1, 2D convolution #1, and aluOp. Each of these three operations can be executed by a different machine thus eliminating the time wasted in processing the same output three times by different operators one after another. With parallelism there will be no delay in passing data along when a computer has completed one operation.

These two examples illustrate the bulk of possibilities that may benefit from parallel processing. They also represent the very parallel nature of the analysis of remotely sensed data.

6 Results and Conclusions

In preliminary testing, using only three computers, no outstanding gain in speed was observed. In most cases the time to execute was almost fifty percent greater than that of using a single computer. As the data volume of remotely sensing images is much larger than that of images of other applications, image transfer among distributed PCs through conventional LAN links might have caused this problem and so no improvement in performance was observed. However, this situation should be solved by installing many one-board PCs in an enclosure through local bus of PCs such as PCI (Peripheral Component Interconnect). The coarse parallel image computing system appears to be very economical and viable. The parallel system is ideally suited to processing remotely sensed data due to the parallel nature of techniques used to analyzed these types of data. Further investigation, with one-board PCs and more computers and more complicated programs, is still required to evaluate the full efficiency of the system.

7 Acknowledgments

The author wishes to express hearty thanks to Dilip A. Ogale, a student from the University of Waterloo, for assistance with programming and experimentation for this study.

8 References

1) Baker, Low and B. J. Smith. *Parallel Programming*. McGraw-Hill, New York, 1996.

2) Lintz Jr., J. and D. S. Simonett. *Remote Sensing of Environment.* Addison Wesley Publishing Company, Reading, Massachusetts, 1976.

3) Iisaka, J. and T. Sakurai-Amano, "Automated Terrain Feature Extraction from Remotely Sensed Images integrating Spectral, Spatial and Geometrical Attributes of Objects", Proc. GIS/LIS'95(1995) Vol.1 pp.486-495.

4) Iisaka, J. "Structural spatial information extraction from remotely sensed data.' Proc. IGARSS(1989), Vancouver, B.C., pp.1224-1227.

5) Iisaka, J. and W. Russell. "Microcomputer based Terrain Understanding and Land Information Processing System", Proc. 7th Thematic Conference on Remote Sensing for Exploration Geology(1989), Calgary, Alberta, pp.939-953.

6) Richard, J. A. *Remote Sensing Digital Image Analysis.* Springer-Verlag, New York, 1986.

7) Thomas, I. L., V.M. Benning and N. Ching. "*Classification of Remotely Sensed* " , Imprint by IOP Publishing, Bristol, England, 1987.

8) WiT Version 4.8.5 Programmer's Manual. Logical Vision Ltd., Burnaby, B.C., 1995.

9) WiT Version 4.8.5 User's Manual. Logical Vision Ltd., Bumaby, B.C.1995

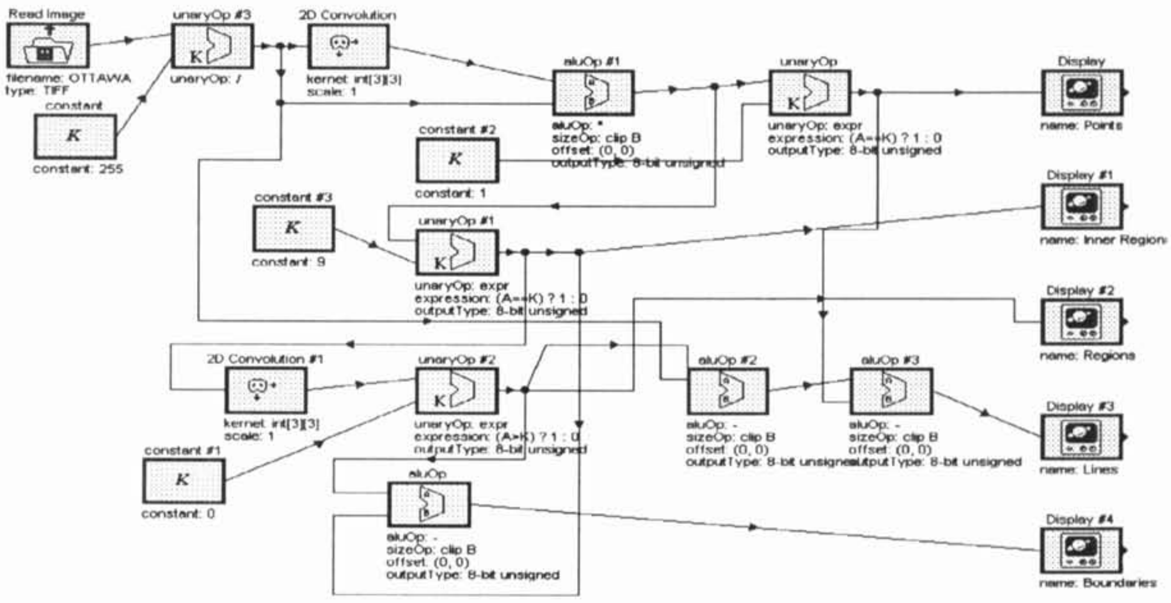

Figure 2: Parallel processing of spatial feature extraction by pixel swapping.