A NEW COMPUTER ARCHITECTURE FOR IMAGE PROCESSING[1]

Wayne A. Davis and De-Lei Lee[2]

Deptartment of Computing Science
University of Alberta
Edmonton, Alberta, T6G 2H1
CANADA

Abstract

This paper describes a new parallel computer architecture for image processing. The proposed architecture is capable of supporting a wide range of image processing tasks that are diverse in their computational requirements by matching the algorithm with the architecture. This versatility is achieved primarily with the use of a new storage scheme in conjunction with a new multistage interconnection network. This organization allows various partitions of the image data such that the data can be accessed in parallel without memory conflict.

INTRODUCTION

Computational vision is a process that requires considerable insight and computation power [8]. It is used in many different applications such as: the processing of medical images for diagnosis and verification; visual inspection of manufactured articles such as printed circuit boards and coins; analysis of satellite and airborne data for determining land use, estimation of crop growth and ice tracking; weather pattern analysis by cloud tracking; etc. The uses are many, diverse and steadily increasing in importance. In some applications, such as blood cell tracking, a real time response is required. It is therefore important that vision systems have sufficient capability to perform exacting tasks quickly and reliably.

This paper presents a new architecture for image processing and computer vision. This new approach permits a wide range of applications to utilize a single system to a significant advantage. Fundamentally the approach is based on a new method of partitioning the data so that it can be accessed in many different ways that are suitable to the specific application.

A multiple-processor architecture of the SIMD type consisting of N processing elements and N memory modules is introduced. The proposed architecture implements a set of N partition strategies for the purpose of algorithm-architecture matching. As a result, dissimilar image processing operations can be implemented with dissimilar constructs on the architecture as opposed to von Neumann computers which implement dissimilar operations with similar constructs. Such an architectural feature is achieved by making use of a novel storage scheme in conjunction with a fast multistage interconnection network. Functionally, the storage scheme defines the set of partition strategies, and the interconnection network moves partitioned parts between processing elements and memory modules. Fundamental properties of the computer architecture are given, and some of the implementation considerations are provided.

BACKGROUND

Current computer techniques for computer vision using sequential algorithms are feasible on only the largest and fastest of machines, which makes their use not always feasible and not always economic[1]. To overcome this problem, researchers have investigated ways of speeding up computation. The initial tactic was to increase the speed of the circuits themselves, and then investigation began on ways of modifying the architecture. The first architectural approach was to utilize pipelining techniques which provide parallel operations on sequential algorithms by performing different operations on the instruction stream in parallel. Basically this is a sequential process, with interleaving so that sequential instructions are overlapped. This provides only a limited speed up factor and is not suitable for massively parallel systems which are needed for some applications. What is needed is what can be called true parallelism where several CPUs actually operate in parallel.

1. This research was supported in part by NSERC Grants A7634 and A9196.
2. D.-L. Lee is currently with the Dept. of Computer Science, York Univ., Toronto.

Several different models and a classification of parallel architectures have been presented [3]. One of these that stands out as being applicable to the class of image processing problems is the Single Instruction Multiple Data (SIMD) model as described by Unger [10]. This paper restricts itself to the SIMD model or some variation of it [1]. The SIMD model is very applicable to problems where the data consists of an array of numbers and the algorithms that are applied are applied consistently to each and every item in the array. The most obvious of these problems is the problem of image analysis or image processing where the array is an array of grey values (or spectral densities in a multispectral image). Most of the algorithms in this application are applied to a local neighborhood or at least to a limited amount of the image data at a time.

Parallel algorithms have been developed for parallel processor arrays which are not only theoretically sound but have also been implemented and used [1,2]. Parallel processing on SIMD and MIMD computers has changed from an emerging to a mature technology [2,7,8]. There are, however, significant problems that continue to plague the users. One of these is the problem of data distribution. Depending upon the algorithm, the distribution of the data over the memory elements connected to the processing elements is substantially different. Each processing element must access data in parallel, which means that the data must be stored in the memory elements such that it can be accessed in parallel in various ways [8].

A NEW ARCHITECTURE

Before getting to deeply into a new approach, an important aspect that must be considered is the question of generality. In other words, how general or how specific is the machine in question? Clearly the amount of generality that can be implemented must be restricted because of the extra expense involved and the penalty paid for the lack of speed. Therefore this paper will be restricted to as general a SIMD processor as possible without digressing to a general purpose processor. All data will be assumed to be rectangular arrays of homogeneous data elements.

Clearly the general problem of an $m \times n$ array of elements are easily processed on an $m \times n$ array of CPUs that are suitably interconnected and controlled. The real problem occurs when a $p \times q$ array of CPUs is used, where p « m and q « n. In this case the problem degenerates to one of data storage and moving the data to the appropriate CPU so that the various arithmetic and logic operations can be performed. Assuming that there is enough memory in the system to store the complete array without duplication, the question arises as to what to put where. Expressed in another way: How is the data to be partitioned so that the most effective use can be made of it?

Assume the computation model shown in Fig. 1, where N, the number of CPUs, is equal to n^2 for some $n > 1$. Furthermore, assume that the image array is also square and that its size is m^2, where $m = a \cdot n$, and a is a positive integer. In addition, assume that the CPU array is interconnected as shown by the 4×4 array in Fig. 2. This means that in order to process an image, each memory element M_i must be capable of storing $m^2/n^2 = a^2$ elements.

Now, for a moment, consider the various ways of using the data so that the appropriate item will be in the most appropriate memory element. There are four main ways that users like to access the image data [1]:

1. Contiguous data sets, so that each $a \times a$ subregion of the original is available at one time.
2. Distributing the data so that elements 1, a + 1, 2a + 1,..., are avaiable simultaneously, and the items are numbered sequentially from left to right and top to bottom.
3. Each row is available at one time.
4. Each column is available at one time.

Each of these partitionings has it strong points and each has its drawbacks. There is however, a technique of distributing the data so that each requirement can be satisfied. To illustrate, consider the 4×4 array given in Fig 3. Suppose that a 2×2 array of CPUs is to be used to process the data. Then the number in the array of Fig 3 represents the number of the CPU in which the data element at that location is stored. Now consider where the elements are stored. Each row is such that the CPUs can access a row of data at a time with one element per CPU. A similar feature holds for each column. Likewise, if the 2×2 subregions are considered, each subregion is available at one time to each CPU, and a similar argument holds for the distributed partition.

The main concern now is to ensure that when the elements of a partition are being processed the data must be accessed so that the appropriate element is available at the correct time. That is the function of the switching network.

It is necessary to show not only that the data distribution technique will apply to larger arrays, but that the switching network will also function efficiently and effectively. For the answeres to both of these questions the reader is referred to [5,6] for all of the details.

CONCLUSIONS

This paper has discussed the problem of parallel processors for computers and has presented a new approach to providing a SIMD parallel processor for image processing. The problems have been discussed and a new solution proposed.

REFERENCES

1. P.E. Danielsson and S. Levialdi, "Computer architectures for pictorial information systems", *Computer*, vol. 14, pp. 53-67, November 1981.
2. M.J.B. Duff and T.J. Fountain, Ed., *Cellular Logic Image Processing*, Academic Press, 1986.
3. M.J. Flynn, "Some computer organizations and their effectiveness", *IEEE Trans. Computer*, vol. C-21, pp. 948-960, September 1972.
4. T.J. Fountain, "An analysis of methods for improving long-range connectivity in meshes", *Lecture Notes in Computer Science*, J. Kittler (Ed.), pp. 259-268, March 1988.
5. De-lei Lee, *A Multiple-processor Architecture for Image Processing*, Ph.D. Thesis, University of Alberta, Mar 1987.
6. De-lei Lee and W.A. Davis, "Performing Global Transforms on a SIMD Machine", *Lecture Notes in Computer Science*, J. Kittler (Ed.), pp. 279-287, March 1988.
7. A. Rosenfeld, "Parallel image processing using cellular arrays", *Computer*, vol. 3, pp. 14-20, January 1983.
8. A. Rosenfeld and A.C. Kak, *Digital Picture Processing*, Academic Press, 1976.
9. H.J. Siegel, "PASM - A partitionable SIMD/MIMD system for image processing and pattern recognition", *IEEE Trans. Computers*, vol. C-30, pp. 934-947, December 1981.
10. S.H. Unger, "A computer oriented toward spatial problems", *Proc. IRE*, vol. 46, pp. 1744-1750, Oct 1958.

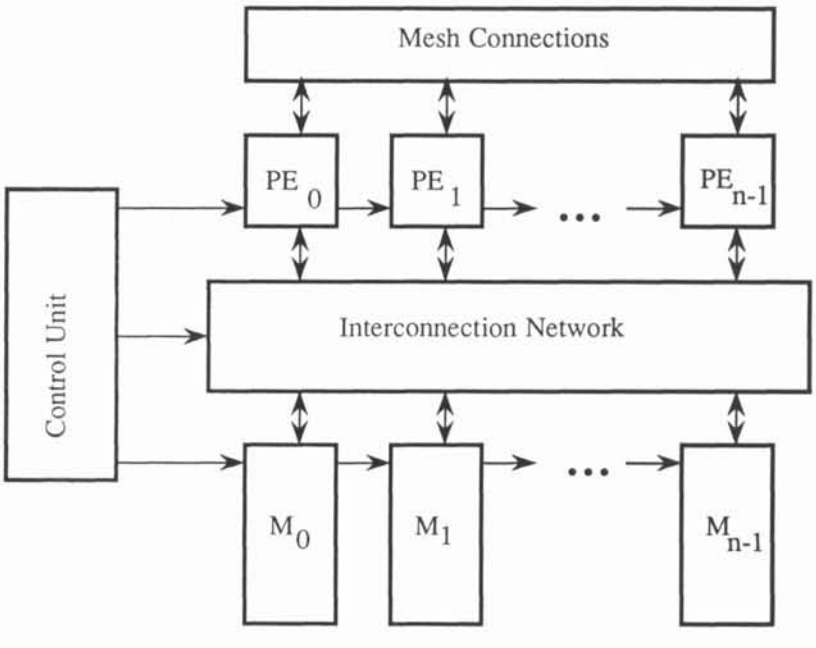

Fig. 1. The Multiple-Processor Configuration

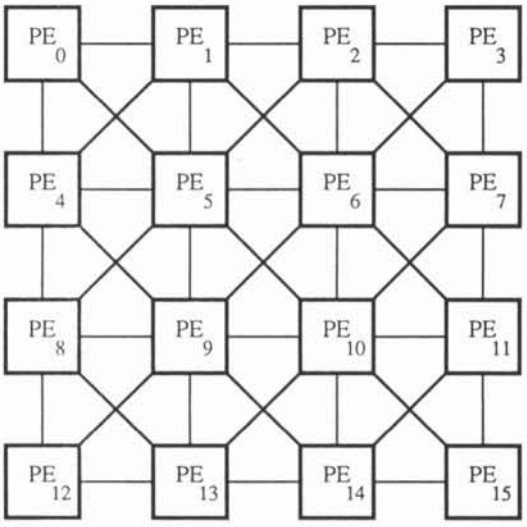

Fig. 2. Processor Connections

$$\begin{vmatrix} 0 & 1 & 2 & 3 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \\ 1 & 0 & 3 & 2 \end{vmatrix}$$

Fig 3. A 4×4 Array.