EFFICIENCY OF FEATURE DEPENDENT ALGORITHMS FOR THE PARALLEL PROCESSING OF IMAGES

by

T. N. Mudge and T. Abdel-Rahman

Computing Research Laboratory
Department of Electrical and Computer Engineering
University of Michigan
Ann Arbor, MI 48109

Abstract--In this paper the concept of feature (in)dependent image processing algorithms is defined. A large class of image processing computers characterized by multiple processor-memory subsystems is efficient when dealing with feature independent algorithms but less efficient when dealing with feature dependent algorithms. Typically such machines are required to perform both types of algorithms. This paper is a preliminary attempt to provide a framework within which to model feature dependent algorithms, and to, for example, quantify the inefficiency that can occur when they are executed on the above type of parallel image processors.

Keywords--feature dependent algorithms, image processing, parallel processing.

1. Introduction

The economics of modern digital integrated circuit technology no longer restricts the designers of digital systems to the classical serial interpreter typified by the von Neumann uniprocessor architecture. This trend away from conventional machines is particularly well developed in the field of image processing where the large data sets (64K bytes to 4M bytes per image) and the high processing rates (near term predictions of 1 to 100 billion operations per second have been made in [1]) make special purpose machines an economic necessity [2]. A number of people have proposed/constructed special purpose machines for image processing. These are surveyed in [3-5].

An architectural characteristic of most of these special purpose image processors is a large number of processors working in parallel. Parallel processing is a natural strategy for dealing with the large data sets and high processing rates encountered in image processing applications; furthermore, the nature of the data and the nature of many of the algorithms make parallel processing particularly attractive. The data is usually a large two dimensional array, and many of the low level image processing algorithms can be decomposed into a large number of concurrent *neighborhood* operations. Examples include: various filtering algorithms such as smoothing to reduce high frequency noise and median filtering to reduce salt-and-pepper noise; edge detection algorithms that use operators such as the Sobel operator and the Hueckel operator; and various coding algorithms such as block truncation coding and cosine transform coding.

A natural architecture for the above class of image processing algorithms is a multiprocessor in which equal subimages are assigned to separate processors for processing. For the purpose of this discussion we will classify such processors as *multiple subimage processors* (MSP's). As might be expected, a large number of the proposed/constructed special purpose image processors can be viewed as MSP's. Figure 1 shows a block diagram of a generic MSP. Subimage *i* is handled by its own processor-memory subsystem, processing element *i* (PE_i). The PE's can communicate through some form of interconnection network (ICN). Specific examples of MSP's include: the proposed PASM architecture [6], which plans to employ multi-path routing-networks to connect a set of 1024 PE's; CLIP4 [7], a 96 × 96 array of simple bit-processors, each with a 32 bit RAM and an ICN that connects nearest neighbors in the array; the Distributed Array Processor [8], a 64 × 64 array of processors with 4K-bit storage per processor and an ICN that connects nearest neighbors in the array and provides a bus per row and column; the Massively Parallel Processor [9], a 128 × 128 array of processors with 1K-bit storage per processor and an ICN that connects nearest neighbors; and the Adaptive Array Processor [10], whose building block is a single chip 8 × 8 array with 96 bits of storage per processor.

In general, MSP's are highly efficient at performing neighborhood operations such as those listed above. These types of operations are an important subclass of what we will term *feature independent* image processing algorithms. Feature independent algorithms are characterized by equal processing per pixel. In other words, each pixel receives the same amount of processing

This work was supported in part by AFOSR grant F49620-82-C-0089.

Preprint from the Proc. of the 1983 Int'l Conf. on Parallel Processing, Aug. 1983, pp. 369-373.

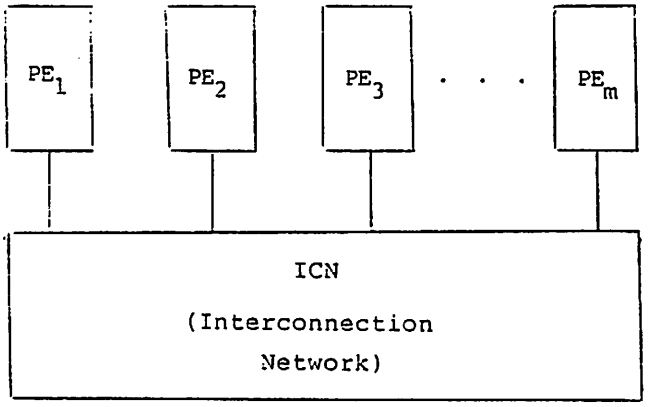


Figure 1. Generic MSP.

regardless of whether or not it is part of a feature of interest such as a line segment. As well as many neighborhood operations there are other algorithms such as histogramming and the Fourier transform which are feature independent. Unlike neighborhood operations these algorithms require significant amounts of data to be moved between processors. The effectiveness of MSP's at performing them is dependent on the bandwidth of the ICN shown in Figure 1. A multiprocessor like PASM with a high bandwidth ICN can perform such algorithms relatively easily [11-13]. Therefore, the concept of a multiprocessor in which equal subimages are assigned to separate processors for processing is also a natural way of handling the complete range of feature independent algorithms, provided the ICN is appropriate for the types of feature independent algorithms anticipated.

Although the above concept is natural for feature dependent algorithms, it becomes less attractive for *feature dependent* image processing algorithms. Feature dependent algorithms are characterized by unequal amounts of processing per pixel. This might arise when a pixel is part of a feature of interest and because of that requires separate treatment. A simple example of a feature dependent algorithm is contour tracing; only edge pixels are involved in the algorithm. In an image processing application the initial sequence of algorithms involves mostly feature independent algorithms because they are concerned with general image enhancement and potential feature location. The subsequent sequence of algorithms is much more likely to involve feature dependent algorithms because specific features are sought from the set of potential locations.

Consider processing an N-pixel image on an MSP machine having m PE's. In normal MSP operation the image is divided into N/m subimages of equal size, and each subimage is processed by a single PE. However, in the case of feature dependent algorithms the image should be divided into subimages of equal *interest*, i.e., subimages having equal numbers of pixels of interest. If, in the case of feature dependent algorithms images are divided into subimages of equal size, some PE's will receive fewer pixels of interest. This uneven distribution of work will result in some PE's being idle during part of the algorithm. Dividing the image into subimages of equal interest requires that the distribution of pixels of interest over the image can be calculated. This is not always possible. On the other hand, it may be possible, but the calculation and the redistribution on the basis of interest may involve more computation than that lost through the inefficiency of having some PE's idle during part of the algorithm.

This paper is a preliminary attempt to provide a framework within which to model feature dependent algorithms, and to, for example, quantify the above inefficiency to assist in decisions about image distribution among PE's.

The following section develops a mathematical model of feature dependent algorithms. Section 3 tests it using some real image data with edge pixels as the pixels of interest. Section 4 concludes the discussion.

2. Mathematical Model of Feature Dependent Algorithms

Consider an N-pixel image and an m-PE MSP system. Assume that the pixels of interest occur randomly in the image and that the probability of a pixel being of interest is p regardless of its position. Assume that the MSP system is executing an image processing algorithm on the image. Let the time to complete the algorithm be a function, f, of the number of pixels of interest in the image, i.e., the algorithm is a feature dependent one.

For the single PE case ($m=1$) the expected value of the execution time, T_1, is given by:

$$T_1 = f(Np) \tag{1}$$

For the m-PE case assume that the image is divided among the m PE's on an equal size basis. Each PE holds an $n = N/m$ pixel subimage. Let X_i to be the random variable describing the number of pixels of interest in subimage i, $i=1,2,...,m$. From the above assumption that the probability of a pixel being of interest is p regardless of its position, it follows that the X_i's are identically independently distributed (i.i.d) random variables with a binomial distribution (see Figure 2).

Let T_{max} be the expected value of the maximum execution time among all PE's. Since the algorithm is not finished until all the m PE's have completed the work in their subimage, it follows that:

$$T_m = f(E[X_{max}]) \tag{2}$$

Where:

$$X_{max} = max(X_1, X_2, ..., X_m) \tag{3}$$

To evaluate T_m consider the following. Let p_j be the probability of exactly j pixels of interest occurring in subimage i:

$$Pr\{X_i = j\} = p_j = \binom{n}{j} p^j (1-p)^{n-j} \tag{4}$$

Let q_j be the probability of greater than j pixels of interest occurring in subimage i:

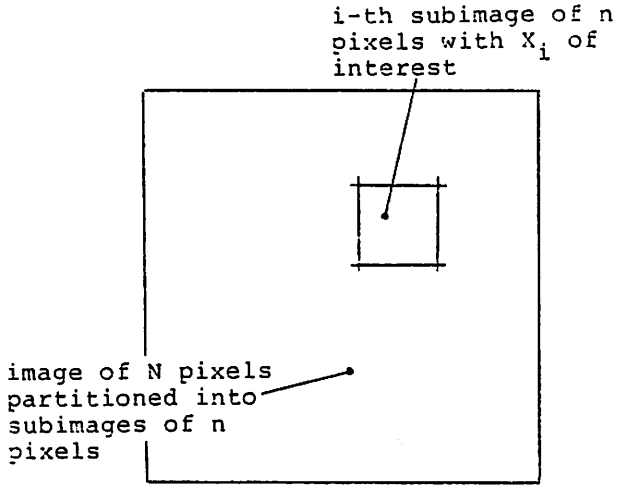


Figure 2. A subimage and its associated random variable.

$$Pr\{X_i > j\} = q_j = \sum_{r=j+1}^{n} p_r \tag{5}$$

Then:

$$q_j = \sum_{r=j+1}^{n} \binom{n}{r} p^r (1-p)^{n-r} \tag{6}$$

Let $P(z)$ be the generating function for the sequence p_j, $j=0,1,...,n$:

$$P(z) = p_0 + p_1 z + ... + p_n z^n \tag{7}$$

Let $Q(z)$ be the generating function for the sequence q_j, $j=0,1,...,n$:

$$Q(z) = q_0 + q_1 z + ... + q_n z^n \tag{8}$$

From (7) and (8) it follows that:

$$Q(z) = \frac{1 - P(z)}{1 - z} \tag{9}$$

Equation (9) can be verified by equating the z coefficients on both sides of the equation:

$$1 - P(z) = (1 - z)Q(z) \qquad (q_n = 0) \tag{10}$$

See [14].
Differentiating $P(z)$ with respect to z yields:

$$P'(z) = p_1 + 2p_2 z + ... + np_n z^{n-1} \tag{11}$$

Evaluating $P'(z)$ at $z=1$ yields:

$$P'(1) = p_1 + 2p_2 + ... + np_n \tag{12}$$

The right hand side of the above equation is simply $E[X_i]$. Thus:

$$E[X_i] = P'(1) \tag{13}$$

Differentiating both sides of (10) yields:

$$-P'(z) = -Q(z) + (1 - z)Q'(z) \tag{14}$$

Evaluating (14) at $z=1$ yields:

$$P'(1) = Q(1) \tag{15}$$

Comparing to (13) gives:

$$E[X_i] = Q(1) \tag{16}$$

Next consider $Pr\{X_{max} \le j\}$:

$$Pr\{X_{max} \le j\} = Pr\{X_1 \le j \text{ and } X_2 \le 2 \cdots \text{ and } X_m \le j\} \tag{17}$$

Since the X_i's are i.i.d, (17) reduces to:

$$Pr\{X_{max} \le j\} = \left[Pr\{X_i \le j\}\right]^m \tag{18}$$

For any i.
Using the relation:

$$Pr\{X_{max} > j\} = 1 - Pr\{X_{max} \le j\} \tag{19}$$

Gives:

$$Pr\{X_{max} > j\} = 1 - \left[Pr\{X_i \le j\}\right]^m \tag{20}$$

But from (16):

$$E[X_{max}] = Q_{max}(1) \tag{21}$$

And, by definition:

$$Q_{max}(z) = Pr\{X_{max} > 1\} + Pr\{X_{max} > 2\} + ... + Pr\{X_{max} > m\} \tag{22}$$

Therefore, substituting (20) into (22) gives:

$$T_m = f\left(\sum_{k=0}^{n} 1 - \left[P\{X_i \le k\}\right]^m\right) \tag{23}$$

Where $Pr\{X_i \le k\}$ is given by:

$$Pr\{X_i \le k\} = \sum_{r=0}^{k} \binom{n}{r} p^r (1-p)^{n-r} \tag{24}$$

Notice tha the value of T_{max} is independent of i because the X_i's are i.i.d.

Following the usual arguments (see [15]) the efficiency E can be defined in terms of T_1 and T_m by:

$$E = \frac{T_1}{m\,T_m} \tag{25}$$

Thus the efficiency of executing feature dependent algorithms can be determined from (1), (23), (24) and f, the function that describes the time to complete the algorithm.

3. Experimental Results

In an attempt to test the above results the following experiment was carried out on a set of images of industrial parts. These images were obtained from the General Motors database for the industrial bin of parts problem [17]. The names of the ones used are listed in Table 1.

The Sobel edge operator was applied to the above images. A pixel was defined to be of interest if and only if it was on an edge. The resulting image was thresholded and the number of edge points (number of pixels of interest) was computed. The threshold value was chosen to give a "good" edge image. All the images are 256x256 with 256 gray levels. The number of pixels of interest in each image and the value of p are also shown in Table 1. The value of p was estimated as the number of pixels of interest divided by the total number of pixels in the image.

The images were divided into subimages of equal sizes and the expected value of the maximum number of pixels was obtained experimentally. The experimental value obtained was compared with its theoretical value obtained from equation (23) with $f=1$, for various values of m. Those results are shown in Graph 2. It can be seen that there is a fairly good agreement between the theoretical results and the experimental results when the the features are edge pixels. The lower of the two curves is the theoretical one. This error is due the our assumption that the probability of a pixel being of interest is not related to its position. In the case of edge pixels this is clearly not so as they cluster in lines. Clustering moves the experimental line higher.

In the case of specific features better results might be obtained if a more accurate stochastic model of the features distribution can be developed. For example, more accurate models of edge pixel distributions have been developed [18], however they apply only to edges and computing T_{max} for them appears to be a problem.

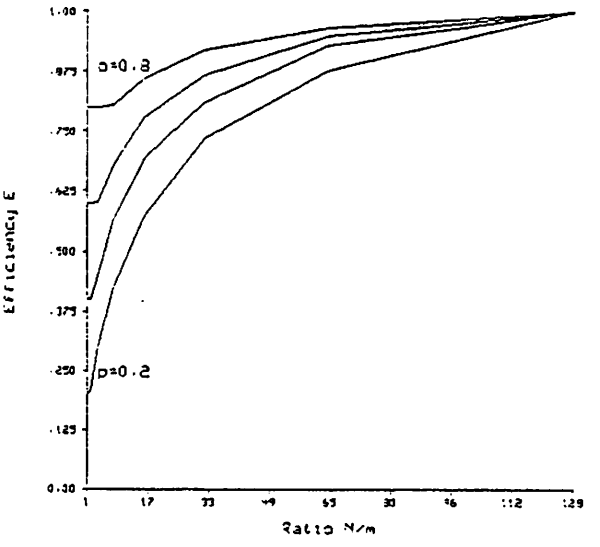
Image Name	No. of Edge Pixels	p
bin1.piv	7732	.118
bin1.piw	12205	.186
bin3.piv	9831	.150
bin5.piz	8032	.123
bin8.piv	5600	.089
yoke1.pit	4421	.064
yoke2.pit	5241	.080
yoke3.pit	8018	.122
rod1.pit	8768	.134
bin1.piw	15822	.241

Table 1.

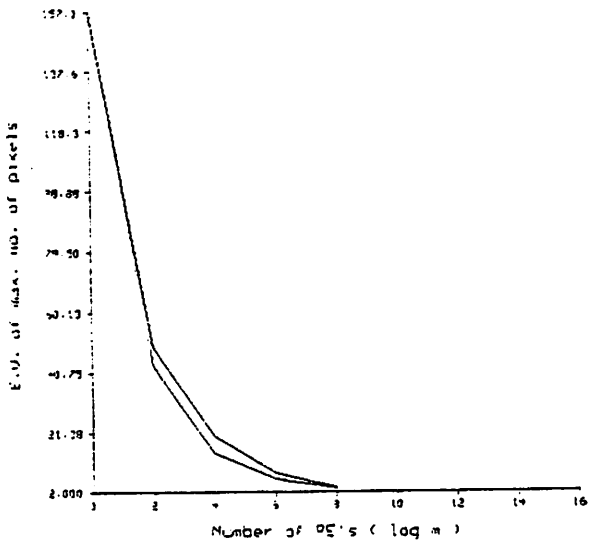
Graph 1 shows the variation of the efficiency, E, as a function of the ratio N/m for p = 0.2, 0.4, 0.6, 0.8.. The graph was plotted by assuming f to be linear. A more realistic function would depend on the specific feature dependent algorithm being considered. However, linear does appear to be a reasonable assumption for a large class of algorithms. For example, a relatively complicated feature dependent algorithm such as the Generalized Hough transform [16] is approximately linear: for each pixel of interest no more than a fixed number of accumulators have to be updated.

If care is taken T_m can be evaluated in $O(n)$ time. The term from (24) should not be evaluated from scratch for each value of k. Also, for large values of n the terms on the right hand side of (24) can be approximated by a Poisson distribution whose terms can in turn be evaluated using Stirling's formula and logarithms.

Several points can be deduced from Graph 1. The efficiency tends to p as N/m goes to 1. This agrees with intuition: if there were as many PE's as pixels, p would be the fraction likely to contain an interesting pixel, and only this fraction would have any work. For very low values of p ($<<0.2$) the efficiency can drop drastically for MSP's processing images that have less than an order of magnitude more pixels than they have PE's. For example, PASM with 1024 PE's will operate at less than 40% efficiency on images of 64 x 64 pixels if p=0.4. On the other hand if the images are 256 x 256 the efficiency jumps to over 80% for the same value of p. Clearly, for high efficiency the image should contain several orders of magnitude more pixels than the MSP has PE's.



Graph 1. E versus N/m



Graph 2.

4. Conclusions

This paper has presented a preliminary attempt to provide a framework within which to model feature dependent algorithms, and to, for example, quantify the inefficiency that can occur in MSP's when subimages of equal size are distributed among the PE's.

The mathematical model was simple enough to allow key terms such as T_{max} to be efficiently computed without compromising the accuracy of the result. Future work might examine how E can be determined if more complex, say Markov, models were used for the features of an image.

5. References

[1] R. Reddy and R. W. Hon, "Computer architectures for vision," in *Computer and Sensor-Based Robots*, G. G. Dodd and L. Rossol (Eds.), Plenum Press, New York, 1979, pp. 169-186.

[2] T. N. Mudge and E. J. Delp, "Special purpose architectures for computer vision," *Proc. of the 15-th Hawaii International Conf. on Systems Science*, Jan. 1982, pp. 378-387.

[3] P. E. Danielsson and S. Levialdi, "Computer architectures for pictorial information systems," *Computer*, vol. 14, no. 11, Nov. 1981, pp. 53-67.

[4] K. Preston, "Cellular logic computers for pattern recognition," *Computer*, vol. 16, no. 1, Jan. 1983, pp. 36-47.

[5] R. A. Rutenbar, T. N. Mudge and D. E. Atkins, "A class of cellular architectures to support physical design automation," *Computing Research Lab. Tech. Report CRL-TR-10-83*, Univ. Michigan, Feb. 1983.

[6] H. J. Siegel, L. J. Siegel, F. C. Kemmerer, P. T. Mueller Jr., H. E. Smalley Jr. and S. D. Smith, "PASM: A partitionable SIMD/MIMD system for image processing and pattern recognition," *IEEE Trans. on Computers*, vol. C-30, no. 12, Dec. 1981, pp. 934-947.

[7] M. J. B. Duff, "Review of the CLIP image processing system," *Proc. National Computer Conf.*, 1978, pp. 1055-1060.

[8] J. K. Iliffe, *Advanced Computer Design*, London: Prentice Hall, Chap. 12, 1982.

[9] K. E. Batcher, "Architecture of a massively parallel processor," *Proc. 7th Annual Symp. on Computer Architecture*, 1980, pp. 168-174.

[10] M. Aoki et al., "An LSI adaptive array processor," *Proc. ISSCC*, Feb. 1982, pp. 122-123.

[11] H. J. Siegel, L. J. Siegel, R. J. McMillan, P. T. Mueller Jr., S. D. Smith, "An SIMD/MIMD multimicroprocessor system for image processing and pattern recognition," *Proc. IEEE Computer Society Conf. on Pattern Recognition and Image Processing*, Aug. 1979.

[12] P. T. Mueller Jr., L. J. Siegel and H. J. Siegel, "Parallel algorithms for the two-dimensional FFT," *Proc. 5-th Int. Conf. on Pattern Recognition*, Dec. 1980.

[13] E. J. Delp, T. N. Mudge, L. J. Siegel and H. J. Siegel, "Parallel processing for computer vision," *Proc. of SPIE-The Int. Society for Optical Engineering*, vol. 336 (Robot Vision), May 1982, pp. 161-167.

[14] W. Feller, *An Introduction to Probability Theory and Its Application*, vol. 1 (3rd edition, revised printing), New York: John Wiley, 1970.

[15] D. J. Kuck, *The Structure of Computers and Computations*, vol. 1, New York: John Wiley, 1978.

[16] D. H. Ballard and C. M. Brown, *Computer Vision*, New Jersey: Prentice-Hall, 1982.

[17] M. L. Baird, "A computer database for the 'Industrial bin of parts problem'," *General Motors Research Lab. Tech. Report GMR-2502*, Aug. 1977.

[18] J. W. Modestino and R. W. Fries, "Construction and properties of a useful two-dimensional random field," *IEEE Trans. on Information Theory*, vol. IT-26, no. 1, Jan. 1980, pp. 44-50.