CASE STUDY OF A PROGRAM FOR THE RECOGNITION OF OCCLUDED PARTS

BY

T. N. Mudge and T. S. Abdel-Rahman
Computing Research Laboratory
Department of Electrical and Computer Engineering
University of Michigan
Ann Arbor, MI 48109

Abstract--A case study is performed on a program for the recognition of occluded industrial parts. The program is running on a VAX 11/780 and is typical of many image processing/pattern recognition programs used in industrial parts recognition. The objective of the study is to investigate the possibility of "parallelizing" the program and then matching it to an appropriate architecture. The nature of the program and the nature of the data processed during each stage of the program are identified to aid selecting the most suitable architecture.

I. Introduction

In computer vision, the problem of matching, in an optimal way, diverse algorithm characteristics to equally diverse multiprocessor architectures is an interesting one. Many factors should be taken into consideration in arriving at the most suitable architecture. The factors include[1,2,3]: algorithm characteristics, processing element design and the type of interconnection network that couples the processing elements.

In order to investigate the impact of the algorithm characteristics on the type of architecture used, a case study is performed on a program for the recognition of occluded parts. Each major stage of the program is identified. The characteristics of the algorithm in each stage are used to determine the most suitable architecture for that stage. To limit the study, only SIMD (Single Instruction stream Multiple Data stream) architectures and MIMD (Multiple Instruction stream Multiple Data stream) architectures are considered. These architectures are described below.

The general configuration of the SIMD architecture is depicted in Figure 1. It consists of a single control unit, N processors, N memory modules and an interconnection network(ICN). Each processor is connected to its own memory module to form a processing element (PE). All processors execute the same set of instructions, hence, there is a single instruction stream. Each processor executes instructions on the set of data stored in its own associated memory module, hence, there is a multiple data stream. The interconnection network facilitates communication between the processing elements.

SIMD architectures are generally more suitable for low level image processing where identical operations are performed on a large number of pixels. The image is partioned into subimages and each processor is assigned a single subimage. Then, executing the same set of instructions, all processors, in parallel, process all subimages of the image.

The general configuration of the MIMD architecture is depicted in Figure 2. It consists of N processors, a shared memory and an interconnection network. Each processor executes its own set of instructions. Hence, there is a multiple instruction stream. Each processor executes its instructions on the data in the shared memory. Hence, there is a multiple data stream. The processors access the shared memory through the interconnection network.

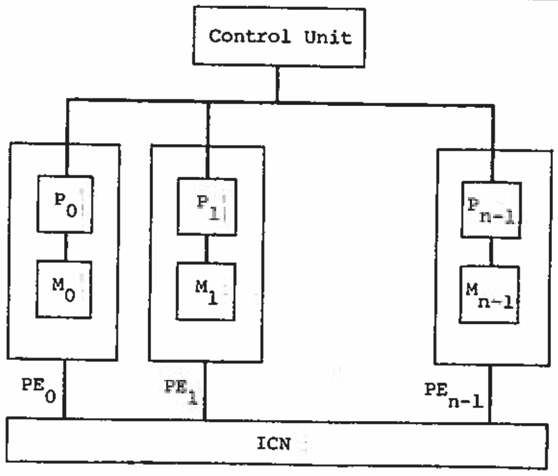

Figure 1. General SIMD Configuration.

MUDGE + RAHMAN ① of ⑤

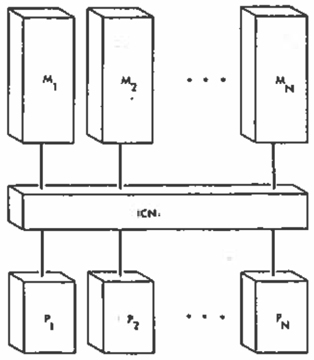

Figure 2. General MIMD Configuration.

MIMD architectures are generally more suitable for high level image analysis. The techniques involved in image analysis are usually referred to as pattern recognition or image classification. Images are generally represented by data structures other than simple two dimensional arrays. Image analysis algorithms include many independent operations on common data that are well suited for MIMD architectures.

II. Program Description

The program under consideration recognizes partially occluded industrial parts by boundary template matching[4]. In industrial part recognition, the type of parts that may appear in scene are always known beforehand. Therefore, industrial parts recognition has much more restricted environment than "natural" scene recognition where little is known about what may appear in the scene. Hence, considerable amount of computation can be saved if this fact is taken into consideration. A database is constructed of all parts that may appear in the application scene. From this database it is possible to determine templates for each part and select a set of features, called salient features, which differentiate between the templates. Those templates are then matched against the application scene using the salient features first. The location and orientation of the best match gives the location and orientation of the object.

The application scene image is first replaced by its edge map. Strengths and slope angles of edge points are extracted from the image. These edge points are then thinned out so only the strongest edge points which maintain boundary continuity remain. The recognition algorithm works with edge boundaries. All images are 256x256 pixels in size with 256 gray levels. The algorithm may be broken down into four major stages: *Edge detection*, *thining*, *linking*, and *subtemplate matching*.

(1) Edge detection

The image is replaced by its edge map using the Frei and Chen edge detection operator[5]. The operator reports the presence of the edge along with its strength and slope angle. The application of the operator mainly involves the convolution of the image with the two 3x3 operator windows shown in figure 3. The first convolution gives the strength of the edge in the *X* direction. The second convolution gives the strength of the edge in the *Y* direction. The strength of the edge as well as its slope angle are obtained form these two quantities.

The input of the operator, the application scene, is a 256x256 image plane of gray level pixels. The output of the operator, edge strengths in the *X* and *Y* directions, is stored in two similar image planes. Although the application of the operator logically reduces the size of the input data from a 256x256 image plane to a much smaller number of edge pixels, the actual representation of the output is still a 256x256 image plane. Hence, there is no reduction in the size of the image data during this stage of the program.

This algorithm is feature independent as each pixel of the input image receives the same amount of processing[6]. The algorithm is context dependent as it requires the values of neighboring pixels. It is a multipass algorithm. It is univariate input as the input consists of only the gray level image. It is multivariate output as the output consists of edge strengths and edge angles.

0	1	0
-1	0	-1
0	1	0

-1	0	1
0	0	0
1	0	-1

Figure 3. The Edge Detector Windows.

This stage of the program is best processed by an SIMD architecture. The image can be divided among the processing elements on an equal size basis without loss of efficiency[6]. Each processing element convolves the operator windows with the portion of the image it holds. The interconnection network is used to transfer boundary pixels needed by neighboring processing elements. As the algorithm is multipass and multivariate, a relatively large local memory is generally required for each processing element.

This stage consumes about 30% of the total execution time of the program on a uniprocessor system.

(2) Thining

The thick edges obtained by the edge detector are thinned out preserving the continuity to obtain the medial line of the part's boundaries in the image. The algorithm described in [7] is used. The algorithm extracts the medial line of a boundary by removing, at each iteration, contour points of the boundary except the ones that may belong to the medial line. Contour points are the points for which at least one of the conditions shown in Figure 4 holds. In order to avoid nonconnected or empty medial lines, each iteration cycle is subdivided into four subcycles. In each subcycle, only those contour points that satisfy just one of the conditions in Figure 4 are removed. That is, in each subcycle, only contour points from one direction are removed. This scheme results in uniform deletion of contour points from all directions. Hence, the algorithm mainly involves convolving the image with several 3x3 windows.

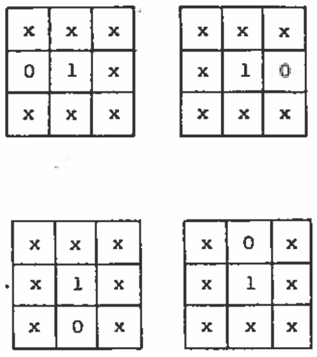

Figure 4. Thinning algorithm Windows

The input of the algorithm is the 256x256 image plane representing the edge strengths. The output of the algorithm is a similar 256x256 image plane containing the thinned edges. There is no reduction in the size of the data during this stage of the program.

The algorithm is feature independent as all pixels are processed equally. It is a single pass algorithm. It is also context dependent as it requires values of neighboring pixels. It is univariate input and univariate output.

This stage of the program is best processed by an SIMD architecture. The image can be divided among the processing elements on an equal size basis without any drop in the efficiency. Each processing element convolves the operator windows with the portion of the image it holds. The interconnection network is used to transfer boundary pixels needed by neighboring processing elements. As the algorithm is single pass and univariate, a relatively smaller size local memory in each processing element is required.

This stage consumes about 15% of the total execution time of the program on a uniprocessor system.

(3) Linking

The thinned edges obtained from the previous stage are linked together to form chains. The angle of each edge element is used to decide if the pixel should belong to the chain or not. If the edge slope angle of the element under consideration is consistent with the directional angle of the chain in the region where the element is located, then the edge element belongs to the chain. Otherwise, the edge element is deleted.

The input of this stage of the program is the 2-D image plane of thinned edges obtained from the previous stage. The output of this stage is a set of linked lists representing the set of chains obtained by linking the edge elements. The elements of the linked list are the *X* and Y coordinates of all pixels belonging to the chain represented by the linked list. As a consequence of this representation of image data, each pixel requires two elements (bytes) to be identified (the *X* and *Y* coordinates) rather than one element (the edge strength or the gray level value). Therefore, depending on the number of linked lists and the length of each list, there may or may not be any reduction in the size of the image data.

This algorithm is feature dependent as pixels are processed according to the value of the angle of the edge element. The algorithm is context dependent. It is also a single pass algorithm. It is univariate input as only the input image pixels are needed. It is, generally, multivariate output as the output consists of a set of linked lists each representing a chain.

If this stage is implemented on an SIMD architecture, and the image is divided among the processing elements on an equal size basis, the efficiency of the system drops[6]. The image should be divided among the processing elements on an equal interest basis. As the algorithm is single pass and univariate, a relatively

MUDGE + RAHMAN 3 of 5

smaller size of local memory is needed on each processing element.

This stage consumes about 5% of the total execution time of the program on a uniprocessor system.

(4) Subtemplate Matching

This is the part recognition stage of the program. Subtemplate sections from the database are matched against application scene edge sections. Subtemplate sections with salient features are used first. In order to simplify the matching process, both the subtemplate section and the edge section are transformed into the $\vartheta - s$ space (slope angle versus arclength representation of data)[8]. The average angle for the template section is determined by averaging the angles of the points in the template section. The average angle for the edge section is determined in a similar fashion. The difference between the two angles is the relative orientation between the template section and the edge section. The template section is then shifted to the same average angle as the edge section and the two sections are compared. If the two sections match, an accumulator at the center of the template is incremented. The amount of increment depends on the degree of match and on the weight of the subtemplate section. This process is repeated for all the subtemplate sections in a template. If there is enough match, i.e., if the accumulator reaches a certain threshold value, then the part represented by the template is recognized and located.

The image data during this stage is represented as a linked list of boundary edge points.

This stage of the program is best processed on an MIMD architecture. Various implementation approachs may be followed. One approach is to let the system processors work together to locate a single part. Each processor is assigned a set of the part's subtemplate sections and attempts to find a match for each subtemplate section assigned to it. Once that section is matched, its edge data is removed from the application scene to speedup the remainder of the matching process. In order to take advantage of the fact that some subtemplate sections have more weight than others, the subtemplate sections are ordered in descending weight and processor *i* is assigned subtemplate section *i mod N*, where *N* is the number of available processors. Consequently, higher weight subtemplate sections are processed first.

Another approach is to let the processors work independently, each processor on a different part template. Each processor attempts to locate the part it is assigned to. Once that part is located, its edge data is removed from the application scene to speedup the remainder of the matching process. If the number of parts exceeds the number of available processors, each processor is assigned more then one part.

A third approach is to attempt to locate all parts in parallel. Subtemplate sections of all parts that may appear in the application scene are ordered in descending weight. Processor *i* is assigned subtemplate sections *i mod N*, where *N* is the total number of available processors. Each processor attempts to match the subtemplate sections assigned to it keeping track of which subtemplate section belongs to which part template. Once a section is matched or a part is located, all edge data belonging to this section or part is removed from the application scene. This speeds up the matching process as high weight subtemplate sections, regardless of which object they belong to, are processed first.

Experiments to determine which of these approaches is most effective are being conducted on a four processor Intel iAPX 432.

This stage consumes about 50% of the total execution time of the program on a uniprocessor system.

III. Summary

The following table summarizes the results obtained:

Program Stage	% Exec Time	Data Type and size	Multiprocessor system			
			Architecture	Local Memory	ICN	Eff.
Edge Detection	30%	2-D array 256x256	SIMD	Larger	Complex	High
Thining	15%	2-D array 256x256	SIMD	Smaller	Complex	High
Linking	5%	2-D array 256x256	SIMD	Smaller	Complex	Drops
Matching	50%	Linked list Appl. Depn.	MIMD	——	Complex	High

Table 1. Summary.

The % execution time values are based on several runs of the program on a uniprocessor VAX 11/780 with varying application scene complexity.

IV. References

[1] Swain, Siegel and El-Achkar, "Multiprocessor Implementation of Image Pattern Recognition:A general Approach," *IEEE Trans. on Pattern Recognition*, Vol.1, pp 309-317.

[illegible]

[2] E. Delp, T. Mudge, L. Siegel and H. Siegel, "Parallel Processing for Computer Vision," *Proceedings of the Society of Photo-optical Instrumentation Engineers Technical Symposium East*, Volume 336 (Robot Vision), May 1982, pp 161-167.

[3] T. Mudge and E. Delp, "Special Purpose Architectures for Computer Vision," *Proceedings of the 15-th Hawaii International Conference on Systems Science*, January 1982, pp 378-387.

[4] J. Turney, T. Mudge, R. Volz and M. Diamond, "Experiments in Occluded Parts Recognition Using the Generalized Hough Transform, *Proceedings of the Conference on Artificial Intelligence*, Okland, MI, April 1983.

[5] W. Frei and C. Chen, "Fast Boundary Detection: A Generalization and a New Algorithm," *IEEE Trans. on Computers*, Vol. C-26, No.10, October 1977, pp 988-998.

[6] T. Mudge and T. Abdel-Rahman, "Efficiency of Feature Dependent Algorithms for the Parallel Processing of Images," *Proc. ICPP, Belliare, Michigan*, August 25-28, 1983, pp 369-373.

[7] R. Stefanelli and A. Rosenfeld, "Some Parallel Thinning Algorithms for Digital Pictures," *J. ACM*, Vol.18, No.2, April 1971, pp 255-264.

[8] W. Perkins, "A Model Based Vision System for Industrial Parts," *IEEE Trans. on Computers*, Vol.C-27, No.2, February 1978, pp 126-143.