Parallel processing for computer vision

Edward J. Delp, T. N. Mudge
University of Michigan, Robotics Research Laboratory
Department of Electrical and Computer Engineering
Ann Arbor, Michigan 48109

Leah J. Siegel, H. J. Siegel
Purdue University, School of Electrical Engineering
West Lafayette, Indiana 47907

Abstract

It has been estimated that processor speeds on the order of 1 to 100 billion operations per second will be required to solve some of the current problems in computer vision. This paper overviews the use of parallel processing techniques for various vision tasks using a parallel processing computer architecture known as PASM (partitionable SIMD/MIMD machine). PASM is a large-scale multimicroprocessor system being designed for image processing and pattern recognition. It can be dynamically reconfigured to operate as one or more independent SIMD (single instruction stream-multiple data stream) and/or MIMD (multiple instruction stream-multiple data stream) machines. This paper begins with a discussion of the computational capabilities required for computer vision. It is then shown how parallel processing, and in particular PASM, can be used to meet these needs.

Introduction

The application of computer vision techniques in industrial robots is an important area of research for sensor-based robots [1]. Computer vision has various applications in parts handling, automatic inspection, and assembly. The objective of computer vision is the development of algorithms that will allow object segmentation based on a set of features relevant to a wide range of applications, supplying real-time sensory feedback information, and functioning effectively in the noisy environments encountered in industrial settings. One distinct problem in computer vision is the large data rates necessary to perform the vision task. The computational requirements of vision algorithms, particularly the high processing rates and the large data sets of predominately two-dimensional arrays, can be best met by a computer architecture that has been tailored to those requirements. This paper examines the use of parallel processing techniques to alleviate the computational problems encountered in computer vision. We examine the use of a multimicroprocessor system in which the vision tasks will be divided among the various processors, the processors will then operate in parallel.

Past work in the computer vision area has resulted in several experimental systems for industrial robots as well as a few cost-effective commercial systems [1,2,3]. The major problems in the design of these systems have been segmentation, object location, object orientation, and object recognition. A typical vision system consists of the following processing steps:

(1) Image acquisition of one or more views of the object area.

(2) Image preprocessing to remove such things as salt-and-pepper noise, filtering to correct uneven lighting, etc.

(3) Object segmentation and feature extraction. The features used include gray-level edges, texture edges, and shape attributes. The features extracted are highly application dependent.

(4) Classification or feature evaluation with respect to object identification, position, and orientation. The classification operation is a higher level operation and is one of the more difficult steps in the vision operation.

A number of articles have appeared in recent years that identify the particular computational needs of computer vision and the related area of image processing [4,6,6]. An important point generally agreed upon is that computer vision algorithms require very high computation rates if they are to be performed in a "reasonable" time. In many applications it is not necessary that the vision algorithms be performed in real-time because of other constraints in the particular application, such as the time required for the robot to execute a "pick-up" task or the time necessary for a conveyor to move an object. The term "relevant time" is often used to denote the time required for the vision task or sub-task so that other processes are not waiting on the vision task to be completed. It has been estimated [6] that processor speeds on the order of 1 to 100 billion operations per second will be required to solve some of the current problems in computer vision in relevant time. Because of these high processing requirements it has been suggested that parallel processing be used to perform the vision operations. In the next sections we shall discuss some properties of vision algorithms and their parallel implementations.

This research was supported by the National Science Foundation under Grants ECS-7909016 and MCS-8009315, by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under Grant No. AFOSR-78-3581, and by the Defense Mapping Agency, monitored by the United States Air Force Rome Air Development Center Information Sciences Division, under Contact No. F3602-81-C-0193 through the Southeastern Center for Electrical Engineering Education. The United States Government is authorized to reproduce and distribute reprints for governmental purposes not-withstanding any copyright notation herein.

Characteristics of Vision Algorithms

In this section various classes of algorithms that are used in computer vision/image processing are discussed. The algorithms will be identified with the various stages of computer vision as outlined in the introduction. In addition, the algorithms will be discussed with regard to the type of data movements and the type of operations they require. Image processing algorithms can be broadly classified into two categories: global operations and neighborhood operations. Global operations involve the use of pixels from the entire image to compute the desired result. Typical examples of these operations are Fourier transforms, shading correction, and shape analysis. Neighborhood operations are based on using pixels within a local neighborhood of the pixel to be modified. The neighborhood is usually a 3x3 window around the pixel. Examples of this type of operation include edge detection, median filtering, and edge thinning. Pixel-by-pixel operations such as histogram modification can be considered as neighborhood operations where the neighborhood is one pixel [7]. There are some operations such as contour tracing and depth from stereo views that can be considered to have components of both kinds of operations.

When using parallel processing systems for computer vision the algorithms can be further divided as being most suitable for one of two types of parallel processing systems: SIMD and MIMD. SIMD (single instruction stream - multiple data stream) machines [8], such as Illiac IV [9] and STARAN [10], typically consist of a set of N processors, N memories, an interconnection network, and a control unit. The control unit broadcasts instructions to the processors, and all enabled ("turned on") processors execute the same instruction at the same time. Each processor executes instructions on a separate data stream. The interconnection network allows interprocessor communication. An MIMD (multiple instruction stream - multiple data stream) machine [8] also typically consists of N processors, N memories, and an interconnection network, but each processor can follow an independent instruction stream (e.g., C.mmp [11] and Cm* [12]). As with SIMD architectures, there is a multiple data stream.

We will now discuss each part of the vision task as outlined in the introduction and mention various algorithms that are used for these tasks. The algorithms discussed are meant to be illustrative of the kind used for vision and are not meant to be exhaustive. The first step in the vision operation is acquisition of views of the object area. The processing at this stage is usually analog (including optical processing) and is beyond the scope of this paper. The reader is referred to [4] for a more detailed discussion of possible processing at this stage.

The second step is that of image preprocessing or enhancement. In many cases the object views are acquired in a noisy environment, which often results in "salt-and-pepper" noise in the image. Representative operations to remove the noise include smoothing and frame averaging. These operations often blur the image. Another operation which has gained popularity in removing "salt-and-pepper" noise *without* blurring the edges is median filtering [13]. Many other noise removal operations are possible [14]. These operations are most often neighborhood operations and ad hoc in nature. Another general class of algorithms used in the preprocessing step is that of lighting correction due to uneven lighting of the object area. Many techniques have been proposed to solve this problem [15,16]. A particular technique is homomorphic filtering which is a global spatial frequency filtering method [17]. Other approaches have been proposed that are simplier in that it is assumed the lighting can be corrected using histogramming followed by smoothing [15].

The third processing step is segmentation and feature extraction. Probably the most representative operation is edge detection. Edge detection can be as simple as pixel thresholding for situations where the object has high contrast relative to the background [1] or as complicated as the Hueckel operator for locating edges with subpixel accuracy [18]. Most edge detectors used are neighborhood operations similar to the Sobel operator [14]. Many of the more advanced edge detectors require floating point operations. Other neighborhood operations used in the feature extraction step include edge thinning, connectivity analysis, and some forms of texture analysis.

Another important set of features that are extracted are shape attributes such as area, perimeter, major and minor axes, the Fourier descriptor, splines, and model based descriptors such as generalized cylinder models [19,20]. These shape attributes are generally global operations. The SRI vision module uses some of the above shape features [1]. Features such as depth or range information are important for the detection of complicated objects and where an object is occluded by another object. Features such as depth, contour tracing, and "blob" analysis are intermediate operations that contain both neighborhood operation components and global operation components.

The final step is classification where the features are evaluated with respect to object identification, location, and orientation. In this step fragmentary information (i.e. the features) about the object is used to make decisions concerning the object. The classification schemes used are usually based on statistical or syntactic pattern recognition [21]. These operations are global in nature and generally require floating point operations.

In Table 1 we summarize computational attributes of various vision algorithms. The "Type of Computation" column indicates whether the algorithm lends itself to SIMD parallelism, or MIMD parallelism. The "Data Movement" column indicates whether the algorithm combines data from local neighborhoods (NBHD's) in the two dimensional image array, or whether it combines data from widely separated points in the image array. Finally, the "Type of Operation" column indicates whether or not the algorithm requires floating point arithmetic. The table is not meant to be definitive: for example, there are algorithms for edge detection requiring global data movements that are not necessarily SIMD. However, the table does show the major algorithmic attributes that vision computations can possess, and that must be considered in the design of any computer that is to execute these computations efficiently.

Computational Attributes of Various Vision Algorithms			
Algorithm	**Type of Computation**	**Data Movement**	**Type of Operation**
Smoothing	SIMD	NBHD	Fixed
Median filtering	SIMD	NBHD	Fixed
Thresholding	SIMD	NBHD	Fixed
Edge growing	MIMD	Intermediate	Fixed
Edge detection	SIMD	NBHD	Fixed/Floating
Edge thinning	SIMD	NBHD	Fixed
Connectivity	SIMD/MIMD	NBHD	Fixed
Contour tracing	MIMD	Intermediate	Fixed
Depth from stereo	MIMD	Intermediate	Floating
Shape analysis	MIMD	Global	Floating
Spline fitting	MIMD	Intermediate	Floating
Classification	MIMD	Global	Fixed/Floating

Overview of PASM

PASM is a partitionable SIMD/MIMD multimicroprocessor system being designed for image processing and pattern recognition applications [24]. It will consist of N processors which can be dynamically reconfigured as one or more independent SIMD and/or MIMD machines of varying sizes, where N is a power of two and may be as large as 1024.

Figure 1 is a block diagram overview of the PASM system. The heart of the PASM system is the Parallel Computation Unit (PCU), which contains N processors, N memory modules, and an interconnection network. The Micro Controllers are a set of microprocessors which act as the control units for the PCU processors in SIMD mode and orchestrate the activities of the PCU processors in MIMD mode. Control Storage contains the programs for the Micro Controllers. The Memory Storage System provides secondary storage for the PCU memory modules. It consists of multiple secondary storage devices connected in a fashion that will allow parallel loading/unloading. The Memory Management System controls the loading and unloading of the PCU memory modules. The System Control Unit is a conventional machine, such as a PDP-11, and is responsible for the overall coordination of the activities of the other components of PASM.

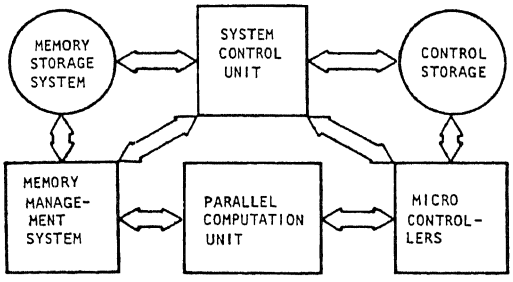

Figure 1. Block Diagram Overview of PASM.

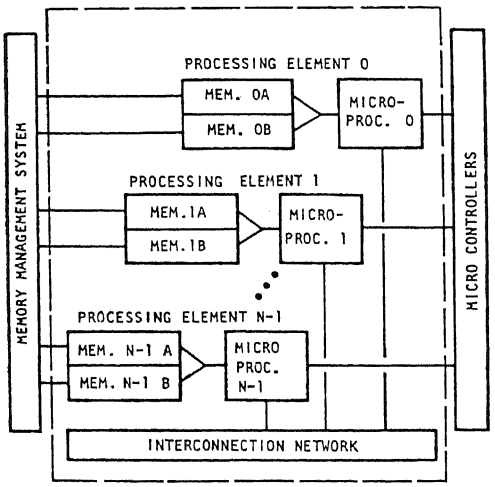

Figure 2. PASM Parallel Computational Unit

The organization of the PCU is shown in Figure 2. The PCU processors are microprocessors that perform the actual SIMD and MIMD computations. The PCU memory modules are used by the PCU processors for data storage in SIMD mode and both data and instruction storage in MIMD mode. A pair of memory units is used for each PCU memory module so that data can be moved between one memory unit and secondary storage while the PCU processor operates on data in the other memory unit (double buffering). A processor and its associated memory module are termed a processing element (PE). The interconnection network provides a means of communication among the PCU's PEs. Either the Augmented Data Manipulator network [22] or the Generalized Cube Network [23] will be used in PASM.

This brief summary of the PASM organization is provided as background for the following sections. For further details of the system architecture, see [22,23,24].

PASM Vision Algorithms

A wide variety of image processing algorithms have been studied for execution on PASM. These include (1) preprocessing/enhancement operations such as smoothing, convolution and filtering, and local area histogram equalization; (2) operations related to feature extraction, such as image correlation for processing of stereo views and FFTs for computation of features such as Fourier descriptors; and (3) operations used as a part of classification, such as global histograms and maximum likelihood classification. The attributes of these parallel algorithms are summarized in this section.

Enchancement Algorithms

Smoothing represents a prototypical neighborhood operation. An SIMD algorithm to smooth an $M \times M$ image is presented in [24,25]. In SIMD algorithm using N PEs, the $N = r^2$ PEs are logically configured as an $r \times r$ grid, on which the $M \times M$ image is superimposed, i.e. each processor holds a $(M/r) \times (M/r)$ subimage. Smoothing is performed on each subimage in parallel, with data transfers needed to smooth the boundary points in each subimage. The number of smoothing operation steps performed with be $(M/r)^2 = M^2/N$. The number of parallel data transfer steps required will be $(4\,M/r + 4)$: each edge of the subimage requires M/r pixels from the adjacent PE, and each corner pixel requires one pixel from a "diagonal" PE. It can be shown [26] that the use of a subimage-per-PE allocation minimizes the data transfers needed. The reduction in arithmetic operations over a serial algorithm is by a factor of N. For realistic values of M and N, $M^2/N >> 4M/\sqrt{N}$, so the time spent in inter-PE communications is negligible compared to the arithmetic processing time. (Other data allocations, e. g., the assignment of complete rows or columns of the image PE, are possible, at a cost of increased data transfers.) The data allocation and inter-PE transfers used for smoothing will be appropriate for a wide variety of local neighborhood operations, including median filtering, some methods of edge detection, and window-based feature analysis [27].

Convolution and filtering also represent local neighborhood operations. SIMD algorithms for general convolution and convolution using rectangular windows have been studied for PASM [26]. Again, subimage-per-PE data allocation is optimal, and the data transfers required are of data (raw pixels or partial results) from the perimeter of each PE's subimage. For convolution of an m^2-point window over and M^2-pixel image, the asymptotic complexity for arithmetic operations is reduced from $O(m^2M^2)$ for the serial algorithm to $O(m^2M^2/N)$ for the N-PE SIMD algorithm. The overhead of inter-PE communications incurred has asymptotic complexity $O(mM/\sqrt{N})$. For convolution with a rectangular window (i.e., summing the points under a window), an $O(M^2)$ serial algorithm can be implemented as an $O(M^2/N)$ SIMD algorithm, with an $O(mM/\sqrt{N})$ overhead.

SIMD algorithms for the enchancement operation of local area histogram equalization and local area brightness and gain control have been developed [28]. Both subimage-per-PE algorithms and algorithms for images arriving in raster format have been considered.

Feature Extraction Algorithms

Feature extraction methods can utilize both the SIMD and MIMD modes of processing on PASM. Both local and global operations may be used.

SIMD algorithms to compute two-dimensional FFTs have been developed [29,30]. Two different algorithm strategies have been explored. The first is a "row-column" approach, in which the two-dimensional DFT of an $L \times M$ array of elements $S(l,m)$ is obtained by computing L one-dimensional M-point transforms on the L rows of the S array to produce an intermediate array G, then computing M one-dimensional L-point transforms on the M columns of the G array. For an $M \times M$ array, this approach can be used with N PEs, $2 \leq N \leq M^2/2$. For $N \leq M$, each PE uses serial one-dimensional FFT algorithms to transform the rows of S and the columns of G. Each PE holds and processes one or more complete rows (columns). The data transfers needed are "uniform shifts," i.e., for all j, PE j transfers data to PE $(j+d)$ mod N for some constant d. For $N > M$, the rows and columns are transformed using SIMD one-dimensional FFTs. Each row (column) of the array is distributed across several PEs. The data transfers needed for the SIMD one-dimensional FFTs resemble FFT butterfly connections on the PEs. Uniform shifts are also needed. In the second approach, a two-dimensional decimation-in-frequency (DIF) algorithm for N PEs, $4 \leq N \leq M^2/4$ is used. The DFT of a $M \times M$ array is obtained recursively in terms of four $(M/2) \times (M/2)$ point DFTs. In order to allocate data, the original image is divided into four quadrants, with each PE holding corresponding points from each quadrant. For both approaches, the reduction in arithmetic operations is by a factor of N, and fewer transfers than arithmetic operations are performed. The row-column method is faster for most values of M and N; the DIF method becomes faster when $N >> M$.

SIMD image correlation algorithms employ the convolution techniques described in the discussion of enhancement algorithms. SIMD approaches to image resampling, which may be required as a preprocessing step to correlation, have been studied [31]. Like correlation, resampling is a local neighborhood operation, calling for subimage allocation of data and transfers to a PEs near neighbors. The SIMD algorithm is based on a formulation of the resampling operation in terms of locally defined offsets, in order to allow the simultaneous calculation of different points in the resampled image space.

Image segmentation is a feature extraction algorithm which lends itself more readily to MIMD execution. Douglass [32] has designed a pipeline architecture for image segmentation based on edge detection and region growing techniques [33]. The method assumes input in raster format, and could easily be run on a partition of PASM.

Classification Algorithms

Like feature extraction, both SIMD and MIMD processing are applicable to classification. For PASM, SIMD algorithms to perform statistical classification and classification-related preprocessing have been studied. Initial studies of MIMD algorithms for syntactic pattern recognition have been conducted.

Histograms are often used to determine thresholds for classification algorithms. SIMD implementations of histogramming have been developed [24,25]. A B-bin histogram of an $M-M$ image can be computed in N PEs, $2 \leq N \leq M^2$. The original image is evenly divided among the PEs, and each PE computes a histogram for its portion of the image. There is no preferred data allocation, as long as all PEs hold approximately the same number of points. A form of overlapped recursive doubling is used to combine the results from the partial histograms. A serial histogram algorithm requires M^2 additions. The N-PE SIMD algorithm uses M^2/N additions, with an overhead of at most $B - 1 + \log_2(N/B)$ transfer/add steps. The algorithm requires communication between pairs of PEs whose numbers (indices) differ only in the i-th bit position for a given value of $i, 0 \leq i < \log_2 N$ (e.g., when $i = 0$, indicating the low order bit position, communication is between PEs 0 and 1, 2 and 3, 4 and 5, etc.)

Maximum likelihood classification is a statistical classification method that processes each pixel of an image independently. Based upon the feature measurements associated with a pixel, its "likelihood" of being a member of each class being used for categorization is calculated. The class for which this likelihood is "maximum" is the class to which the pixel is assigned. Since each pixel is classified independently, the pixels can be evenly divided among the PEs of PASM for processing, speeding up the execution of the task by a factor of N [34]. No inter-PE communication is necessary.

Contextual classifiers [35] are being developed as a method to exploit the spatial/spectral context of a pixel to achieve more accurate classification. In contrast to the maximum likelihood classifier, where each pixel is processed independently, here a pixel's neighboring pixels are also examined in order to help improve classification accuracy. For example, in remote sensing applications, certain classes of ground cover are likely to occur in the "context" of others. This methodology can be applied, in certain cases, to computer vision tasks. However, the computations involved in a statistical contextual classification are quite time consuming. The algorithm complexity (in the worst case) is $O(IJC^P)$ floating point multiplications, where the image is $I \times J$ pixels, C is the number of classes, and P is the size (number of pixels) of the neighborhood. The PASM implementation in [35] reduces this to $O(IJC^P/N)$, at a cost of only $((2(I + J)/\sqrt{N}) + 4)\ C$ inter-PE data transfers. Thus, for $N = 1024$, nearly three orders of magnitude improvement can be attained. Data allocation depends on the size and shape of the neighborhood. For "linear" neighborhoods (e.g., a 1 x 3 window), "stripes" (rows, columns, or diagonals) of the image are allocated to each PE. For "non-linear" neighborhoods such as 3 x 3 windows, rectangular subimages are allocated to each PE. In these cases, transfers would involve neighboring PEs.

Initial studies of the implementation of certain syntactic pattern recognition tasks using parallelism have also been conducted [36]. For these tasks PASM would operate in MIMD mode. Many different approaches to parallel structuring of the problem and allocating the data among the processors are possible, and further study is needed to determine the speedup that can be expected through the use of a system such as PASM.

Summary

In this section, the PASM implementation of various image processing and pattern recognition tasks related to computer vision was discussed. In all cases, the inter-PE data transfers needed could be accomplished by either network proposed for PASM. An advantage of PASM over very specialized hardware (such as a convolution chip) is that a variety of different tasks can be performed using the same system. Furthermore, algorithms and image data sizes can be changed without necessitating hardware changes. Thus, although the overall organization of PASM is being specialized for image processing and pattern recognition, it is not limited to performance of just a single task or algorithm

PASM itself may not be the most efficient multimicroprocessor system for performing a particular computer vision application (i.e., a sequence of particular tasks). However, its flexibility makes it an excellent test bed for experimenting with parallel algorithms for the various tasks that may be part of the specific application, leading to the development of a more efficient, customized multiprocessor system.

Conclusions and Future Research

The algorithms presented in the previous section can all be characterized as "feature-invariant" in the sense that the number of operations they require is independent of the number of features in the image; typically the number of operations is simply a function of the number of pixels in the image and, in some cases, the size of a fixed window. Because of this, feature-invariant algorithms are especially suited to SIMD algorithms. Future research involves examining algorithms for vision that are "feature-dependent" in the sense that the number of operations they require is a function of the number of features in the image. Examples are: spline fitting, and most forms of shape description. These algorithms are suited to an MIMD environment.

A further area of research is in the efficient design of a complete computer vision task, where a task refers to a collection of algorithms. The important issue here is the interface between the algorithms, specifically the way in which data is passed between the algorithms. Inefficiencies can be introduced by having to rearrange data between algorithms. In cases where this rearrangement is unavoidable it may be more efficient, from the task-level view, to use less efficient algorithms that interface more simply.

The initial work described here indicates that parallel processing can provide the significant speed-ups needed for a number of operations which are a part of a computer vision task. Based on this evidence, research on parallel vision systems is being continued, with a focus on complete vision tasks.

References

[1] G. J. Agin, "Computer Vision Systems for Industrial Inspection and Assembly," *IEEE Computer*, Vol. 13, May 1980, pp. 11-20.

[2] S.W. Holland, et al., "A Vision Controlled Robot for Part Transfer," *Research Publication GMR-3120*, General Motors Research Laboratories, Warren, Michigan, 1980.

[3] E. J. Lerner, "Computers That See," *IEEE Spectrum*, Vol. 17, Oct. 1980.

[4] T. N. Mudge and E. J. Delp, "Special Purpose Architectures for Computer Vision," *Proceedings Fifteenth Hawaii Int. Conf. System Science*, Jan. 1982, pp. 378-387.

[5] A. P. Reeves, "Parallel Computer Architectures for Image Processing," *Proceedings of the 1981 Int'l. Conf. on Parallel Processing*, Aug., 1981, pp. 199-206.

[6] R. Reddy, and R. W. Hon, "Computer Architectures for Vision," in *Computer and Sensor-Based Robots*, Eds. G. G. Dodd, and L. Rossol, Plenum Press, New York, 1979, pp. 169-186.

[7] P. E. Danielsson and S. Levialdi, "Computer Architectures for Pictorial Information Systems," *IEEE Computer*, Vol. 14, Nov. 1981, pp. 53-67.

[8] M. J. Flynn, "Very High-Speed Computing System," *Proceedings of the IEEE*, Vol. 54, Dec. 1966, pp. 1901-1909.

[9] W. J. Bouknight, et al., "The Illiac IV System," *Proceedings of the IEEE*, Vol. 60, April 1972, pp. 369-388.

[10] K. E. Batcher, "STARAN Parallel Processor System Hardware," *Proceedings of AFIPS 1977 National Computer Conference*, Vol. 43, May 1974, pp. 405-410.

[11] W. A. Wulf and C. G. Bell, "C.mmp - A Multi-Miniprocessor," *Proceedings of AFIPS 1972 Fall Joint Computer Conference*, Dec. 1972, pp. 765-777.

[12] R. J. Swan, S. H. Fuller, and D. Siewiorek, "Cm*: A Modular Multi-Microprocessor," *Proceedings of AFIPS 1977 National Computer Conference*, June 1977, pp. 637-644.

[13] T. S. Huang, G. J. Yang, and G. Y. Tang, "A Fast Two-Dimensional Median Filtering Algorithm," *IEEE Trans. on ASSP*, Vol. ASSP-27, Feb. 1979, pp. 13-18.

[14] W. K. Pratt, *Digital Image Processing*, New York, John Wiley, 1978.

[15] H. G. Barrow, and J. M. Tenenbaum, "Computational Vision," *Proceedings of the IEEE*, Vol. 69, May 1981, pp. 572-595.

[16] R. P. Kruger and W. B. Thompson, "A Technical and Economic Assessment of Computer Vision for Industrial Inspection and Robotic Assembly," *Proceedings of the IEEE*, Vol. 69, Dec. 1981, pp. 1524-1538.

[17] A. V. Oppenhiem, R. W. Schafer, and T. G. Stockham, "Nonlinear Filtering of Multiplied and Convolved Signals", *Proceedings of the IEEE*, Vol. 56, July 1968, pp. 1264-1291.

[18] M. H. Hueckel, "An Operator Which Locates Edges in Digitized Pictures," *JACM*, Vol. 18, Jan. 1971, pp. 113-125.

[19] W. A. Perkins, "A Model-Based Vision System for Industrial Parts," *IEEE Trans. on Computers*, Vol. C-27, Feb. 1981, pp. 126-143.

[20] T. P. Wallace, O. R. Mitchell, and K. Fukunaga, "Three-Dimensional Shape Analysis Using Local Shape Descriptors," *IEEE Trans on Pattern Analysis and Machine Intelligence*, Vol. PAMI-3, May 1981, pp. 310-323.

[21] A. Rosenfeld, "Image Pattern Recognition," *Proceedings of the IEEE*, Vol. 69, May 1981, pp. 596-605.

[22] H.J. Siegel and R.J. McMillen, "Using the Augmented Data Manipulator Network in PASM." *IEEE Computer*, Vol. 14, Feb. 1981, pp. 25-33.

[23] H.J. Siegel and R.J. McMillen, "The Multistage Cube: A Versatile Interconnection Network," *IEEE Computer*, Vol. 14, Dec. 1981, pp. 65-76.

[24] H. J. Siegel, L. J. Siegel, F. C. Kemmerer, P. T. Muller, H. E. Smalley, and S. D. Smith, "PASM: A Partitionable SIMD/MIMD System for Image Processing and Pattern Recognition," *IEEE Trans. on Computers*, Vol. C-30, Dec. 1981, pp. 934-947.

[25] L.J. Siegel, "Image Processing on a Partitionable SIMD machine," in *Languages and Architectures for Image Processing*, M.J.B. Duff and S. Levialdi, eds., Academic Press, London, 1981, pp. 293-300.

[26] L.J. Siegel, H.J. Siegel, and A.E. Feather, "Parallel Processing Approaches to Image Correlation," *IEEE Trans. on Computers*, Vol. 31, Mar. 1982, pp. 208-218.

[27] P.H. Swain, H.J. Siegel and J. El-Achkar, "Multiprocessor Implementation of Image Pattern Recognition: A General Approach," *Proceedings of 5th Int. Conf. Pattern Recognition*, Dec. 1980, pp. 309-317.

[28] H. J. Siegel, L. J. Siegel, P. H. Swain, et al., "Parallel Image Processing/Feature Extraction Algorithms and Architecture Emulation: Interim Report for Fiscal 1980, Vol. I: Algorithms," Purdue Univ., Elect. Eng. Tech. Report, TR-EE 80-57, Oct. 1980.

[29] L.J. Siegel, P.T. Mueller, Jr., and H.J. Siegel, "FFT Algorithms for SIMD Machines," *Proceedings of Seventeenth Allerton Conf. Communication, Control, and Computing*, Oct. 1979, pp. 1006-1015.

[30] P.T. Mueller, Jr., L.J. Siegel, and H.J. Siegel, "Parallel Algorithms for the Two-Dimensional FFT," *Proceedings of 5th Int. Conf. Pattern Recognition*, Dec. 1980, pp. 497-502.

[31] M.R. Warpenburg and L.J. Siegel, "Image Resampling in an SIMD Environment, *Proceedings of IEEE Comp. Soc. Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, Nov. 1981, pp. 67-75.

[32] R.J. Douglass, "A Pipeline Architecture for Image Segmentation, *Proceedings of Fifteenth Hawaii Int. Conf. System Science*, Jan. 1982, pp. 360-367.

[33] Y. Yakimovsky, " Boundary and Object Detection in Real-World Images," *JAMC*, Vol. 23, 1976.

[34] L. J. Siegel, H. J. Siegel, P. H. Swain, et al., "Parallel Image Processing/Feature Extraction Algorithms and Architecture Emulation: Interim Report for Fiscal 1981, Vol. I: Algorithms," Purdue University, Electrical Engineering Technical Report, TR-EE-81-35, Oct. 1981.

[35] H.J. Siegel, P.H. Swain, and B. W. Smith "Remote Sensing on PASM and CDC Flexible Processors," *Multi-Computer Algorithms and Image Processing*, K. Preston and L. Uhr, eds. Academic Press, New York, NY, 1982.

[36] N.S. Chang and K.S. Fu, "Parallel Parsing of Tree Languages for Syntactic Pattern Recognition," *Proceedings of 1978 IEEE Computer Soc. Conf. Pattern Recognition and Image Processing*, May 1978, pp. 262-268.