Pattern Recognition, Vol. 22, No. 6, pp. 641–652, 1989
Printed in Great Britain

0031-3203/89 $3.00 + .00
Pergamon Press plc
Pattern Recognition Society

PARALLEL ALGORITHMS FOR RECOGNIZING HANDWRITTEN CHARACTERS USING SHAPE FEATURES*

H. F. LI, R. JAYAKUMAR and M. YOUSSEF
Department of Computer Science, Concordia University, Montreal, Quebec, Canada

(*Received* 26 *July* 1988; *received in revised form* 15 *December* 1988; *received for publication* 5 *January* 1989)

Abstract—Systolic algorithms suitable for VLSI implementation for recognizing handwritten characters using shape features are presented. Local shape features, namely start and end points, edge types and their join-relations in the contours of a given character, are first extracted using a systolic algorithm. The global features consisting of the actual sequence of feature points in the contours are then constructed from the local features using a novel systolic ranking algorithm designed for randomized linked lists. By training the algorithms on a set of 1200 handwritten numerals (120 per digit) a classification scheme is developed. A simple PLA like classifier is also presented. Handwritten numerals are recognized using a horizontal, a vertical and a diagonal scan. The scheme works well even if the images are rotated by an angle between −15 and +15 degrees.

Systolic algorithm Architecture Handwritten character Numeral Contour Global features

1. INTRODUCTION

Recent advances in VLSI technology has made possible direct hardware implementation of a number of image processing and pattern recognition functions, with the advantage of improved processing rates. Examples of such VLSI implementations include convolution,[1] minimum-distance classification[2] and context-free language recognition,[3] among others. The first step in developing such special purpose hardware is to design parallel algorithms suitable for VLSI implementation. Design of such VLSI algorithms imposes a very different set of constraints from those imposed by conventional sequential algorithms. In the VLSI domain, it is required to use simple and regular designs, have a high degree of multiprocessing and pipelining, reduce the communication cost by enforcing locality of communicating partners and reduce the input/output pin requirement. Systolic algorithms[4,5] satisfy all these constraints. The homogeneity of the processing elements and the regularity and locality of the interconnection network make it especially suitable for VLSI implementation. In this paper, systolic algorithms for recognizing handwritten characters using shape features are proposed.

Various techniques for recognizing handwritten characters have been reported in the literature. These techniques use either global and statistical features,[6,7] or geometrical and structural features.[8–10] The extracted features should on one hand provide sufficient information for recognition purposes and on the other hand, hide irrelevant detail. In handwritten character recognition, it is preferable to use structural features that can capture the essential characteristics of a shape without worrying about variations in character size, stroke width, and writer style. Shape features like end points, edge types, and cavities have been used by many researchers.[8–11] These features are less sensitive to rotation and are more amenable to natural variations in handwritten characters. Hence shape features are used in this paper, following the approach proposed by Ahmed and Suen.[8,11]

The approach for handwritten character recognition is briefly explained in Section 2 and the features used in the recognition process are identified. These features include local features such as start points, end points and edge types, and global features such as contours. In Section 3 a systolic algorithm to extract the local features is presented. Section 4 describes a novel systolic algorithm to construct the global features from the extracted local features. These systolic algorithms may extract more features than desirable. Finally, in Section 5, the classification scheme using the extracted shape features is presented. The simulation experiment from which the classifier was trained is also explained.

2. AN APPROACH FOR HANDWRITTEN CHARACTER RECOGNITION USING SHAPE FEATURES

Consider an image pattern of binary pixels having M rows and M columns. A pixel can be either white

* This work was supported by the Natural Sciences and Engineering Research Council of Canada under research grants A0921 and A0904.

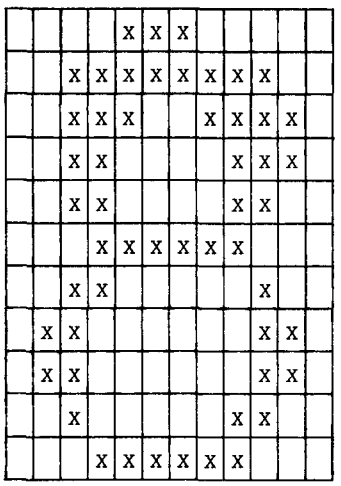


Fig. 1(a). A binary pattern of pixels.

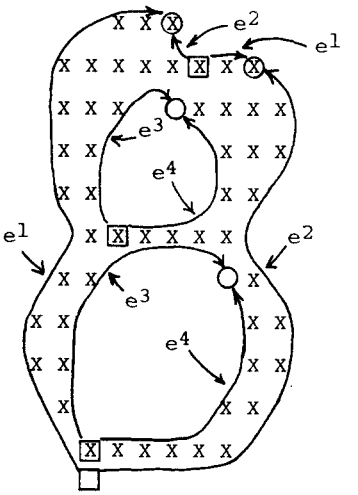


Fig. 1(b). Set of edges surrounding the character of Fig. 1(a).

or dark and the dark pixels constitute a character. The outer and inner (if any) contours of a character can be viewed as a set of edges where an edge corresponds to a transition from a white region into a dark region, or vice versa, while scanning the pattern in a fixed direction. For example, for the pattern in Fig. 1(a), the edges surrounding the character are shown in Fig. 1(b). In the following, it is assumed that the pattern is scanned horizontally from right to left. As shown in Fig. 1(b), each edge possesses a start point and an end point and belongs to one of the following four types.

(i) An outer left edge (e^1) occurs along a contour whose transition is from a dark region into a white region.

(ii) An outer right edge (e^2) occurs along a contour whose transition is from a white region into a dark region.

(iii) An inner left edge (e^3) occurs along a contour whose transition is from a white region into a dark region.

(iv) An inner right edge (e^4) occurs along a contour whose transition is from a dark region into a white region.

Two adjacent edges on a contour have either a common start point or a common end point (or both). Depending on the types of edges meeting at a common start or end point, ten different binary relations (R_1 to R_{10}) can be defined, as summarised in Fig. 2. For example, when an e^1 edge and an e^2 edge share a common start point, relation R_1 is formed between them, denoted also by $e^1 \times e^2$.[12] Similar start and end point relations can be deduced, with $\times$ representing the former and $-$ representing the latter in Fig. 2(a). Ahmed and Suen[11] have shown that these relations are useful in detecting shape features in a character. For example, relations R_7 and R_8 together (abbreviated by R_{11}) indicates the presence of a simple hole in a contour as illustrated in Fig. 2(b). Similarly relations R_1 and R_2 together (abbreviated by R_{12}) imply the contour is the outer contour of some shape. The following implications can likewise be stated.

(i) Relation R_1 implies the beginning of a dark region in a pattern when the pattern is scanned from top to bottom.

(ii) Relations R_2, R_3, R_9 and R_{10} each imply the end of a dark region.

(iii) Relations R_4, R_5, R_6 and R_8 each imply the existence of a simple cavity open at the top.

(iv) Relation R_7 implies the existence of a cavity open at the bottom.

As examples, the relations of the character in Fig. 1 are listed in Fig. 3. From these feature points, edges and relations, it can be deduced that the character in Fig. 1 has an outer contour and two simple holes. These shape features can further be distinguished by their relative positions. This is done by assigning ranks to the features such that a feature in row i of the pattern has the lower rank compared to another in row j if and only if $i < j$ (rows are numbered from top to bottom and columns are numbered from left to right).

In general, an inner (outer) contour of a character enclosing a white (dark) region may contain more than two edges. For the purpose of approximate shape recognition, such a contour can be characterized by the chain of start and end points, termed feature points, together with the attributes associated with them. The attributes we have identified already include the rank and the edge relation at each feature point.

The contour characterization is useful in accommodating variations in writer style in handwritten character recognition as it does not keep track of exact contours but retains critical attributes. For example, a simple analysis will reveal that a contour enclosing a hole (white region) necessarily should contain a start point with the R_7 relation and whose rank is lower than any other start point with the R_1 relation in the

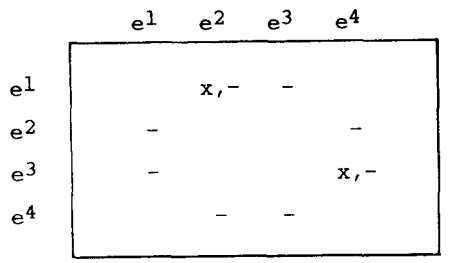


Fig. 2(a). Ten possible relations.

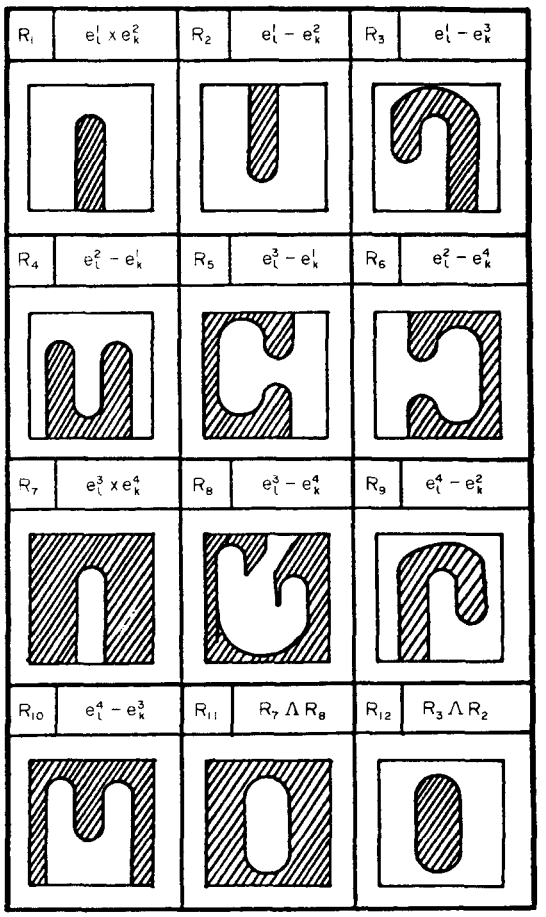


Fig. 2(b). Table showing the twelve relations.

Feature Point	Type of Meeting Edges	Relation
S_1	e^1, e^2	R_1
S_2	e^3, e^4	R_7
E_1	e^3, e^4	R_8
S_3	e^3, e^4	R_7
E_2	e^3, e^4	R_8
E_3	e^1, e^2	R_2

Fig. 3. Feature points and relations in Fig. 1.

chain. Conversely, a contour is enclosing a hole if the start point with the lowest rank in the chain has the R_7 relation. Thus chain 2 in Fig. 4 implies a hole whereas chain 1 does not. Similarly, a chain implies the outer contour of a dark region if the lowest ranked start point in the chain has the R_1 relation. In addition, the rank of a hole can be defined by the rank of the lowest start point in the chain. In the following sections, systolic algorithms to extract and classify these shape features are presented.

3. SYSTOLIC EXTRACTION OF FEATURE POINTS

The preliminary description in the previous section reveals that the recognizer we wish to design is decomposable into three main sections: (a) extraction of feature points and the edge relations at these points with ranks, (b) collection of these features into ordered chains, one for each contour, and (c) classification of these higher order chain features to recognize distinct handwritten numerals/characters. In the sequel, the approach will be formalised and algorithms presented.

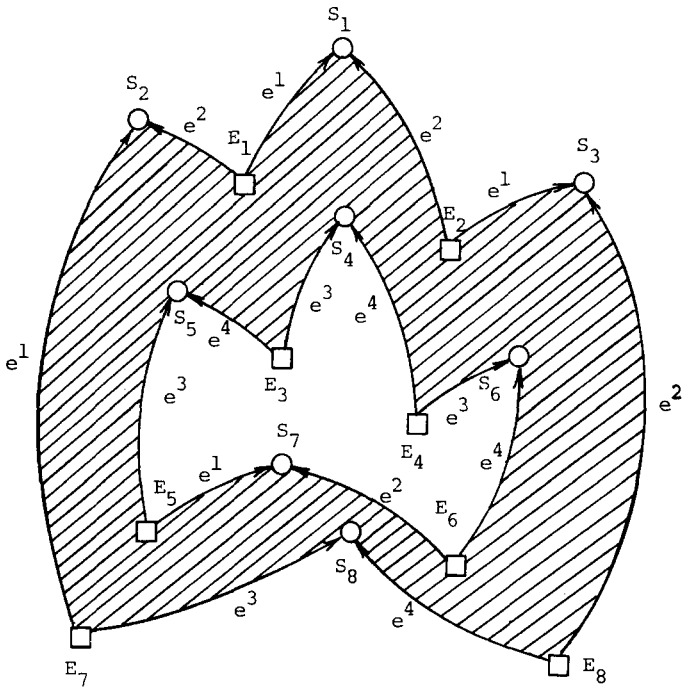
The following definitions apply in characterising the feature extraction, assuming the image is scanned horizontally. Similar definitions can be adopted if the image is scanned in other directions, but are not developed here to avoid confusion.

Definitions. An *edge pixel* is a pixel in the binary image at which a transition from white to dark (or vice versa) has occurred during a scan. The *neighbourhood above* a pixel x includes the pixel y right above x and the left and right neighbouring pixels of y. A *start point* is an edge pixel whose neighbourhood above does not contain an edge pixel of the same type of transition. Thus a start point at coordinate (i,j) implies the absence of a similar edge pixel in coordinates $(i-1,j-1)$, $(i-1,j)$ and $(i-1,j+1)$. A contour enclosing a dark or white region is therefore representable by a sequence of start points on the contour. On the contour segment between two successive start points, an end point is identified by using one of the highest ranked pixels in the segment. However, this natural definition of an end point is slightly modified to accommodate our systolic design and we use an *end point* for the edge segment between two successive start points in a contour as one of the highest ranked non-edge pixel(s) which are adjacent to an edge pixel on the contour segment. A contour is traversed in a fixed direction. A contour enclosing a white region is traversed clockwise whereas that enclosing a dark region is traversed in a counter-clockwise direction. For each end point x in a contour, the start point immediately preceding x in the direction of traversal is called the *predecessor* of x, and the start point immediately following x in the direction of traversal is called the *successor* of x.

With these definitions, the problem of systolic feature extraction can now be stated abstractly.

Input. A binary image pattern, one column per cycle.

Output. For each contour in the image pattern, the



Chain 1: (S_1, E_2, S_3, E_8, S_8, E_7, S_2, E_1)
Chain 2: (S_4, E_4, S_6, E_6, S_7, E_5, S_5, E_3)

Fig. 4. Chains surrounding a pattern.

pixels on the contour are visited* and all start and end points in each row are detected. For each end point x, an end point token is generated and this token identifies the predecessor and successor of x, the edge relation (R_1 to R_{10}) and the coordinate at x.

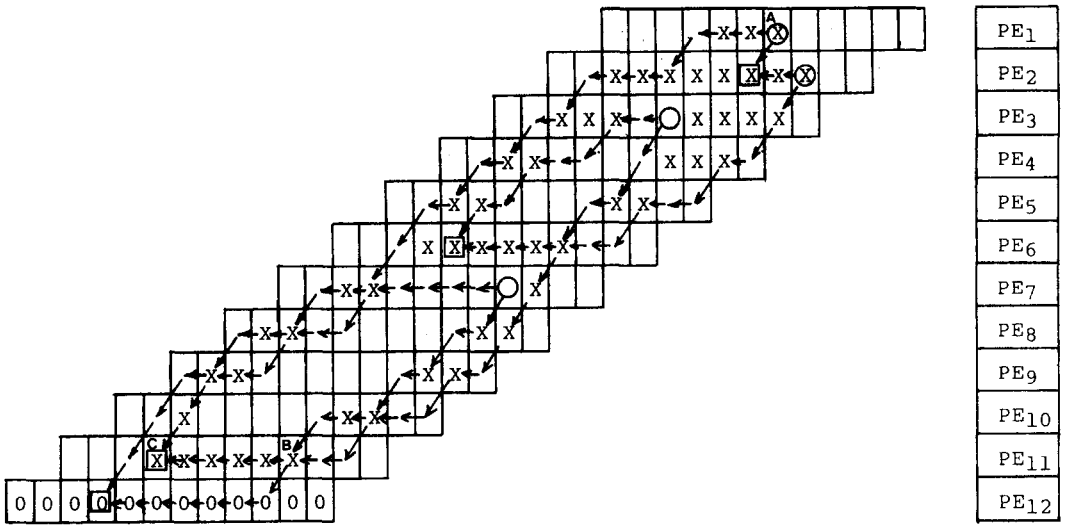
The above problem can be easily solved by a sequential contour tracing algorithm. However, to allow systolic data flow in which the image pixels migrate through a systolic processor in a single pass and internal processing elements only communicate locally, the problem becomes nontrivial. We will first give a concise account of the systolic algorithm before giving a formal description and proof of correctness.

The systolic processor consists of a column of $M + 1$ processing elements (PE) to process a pattern with M rows. The extra PE is needed to account for proper termination in case the last row contains some contour so that PE_{M+1} will hold the corresponding end point(s). The PE's are indexed from top to bottom, PE_1 being the topmost, as shown in Fig. 5. In general, PE_i receives and processes one pixel in row i of the image every cycle; PE_{M+1} always receives white pixels (indicated as 0's in the image in Fig. 5) for "rounding" off an image.

Constrained by the systolic data flow of one pixel per cycle and the need to detect all start points, the image pattern is skewed so that PE_i will process pixel (i,j) at cycle $t + 2$ if PE_{i-1} processes pixel $(i - 1,j)$ at cycle t. This is achieved by either sending in the original image with the above skew or passing the ith row of the image through a row of $2(i - 1)$ delays. For the pattern in Fig. 1(b), the skewed pattern is as shown in Fig. 5. With this skewing, if PE_i detects an edge pixel in a certain cycle but is unaware of any such pixel detected by PE_{i-1} in the last three cycles, it can be verified that the edge pixel is a start point as its neighbourhood above does not contain a similar edge pixel.

Upon detecting a start point, PE_i generates two edge tokens. Each edge token carries information regarding the start point and edge type. A modulo-M counter inside each PE will provide coordinate information in generating the token. One of the edge tokens (right edge) is passed down to PE_{i+1} in the following cycle while the other one (left edge) is kept in PE_i to continue the contour tracing. In Fig. 5, the former token corresponds to a south-west arrow from a start point to its next row neighbour and the latter token corresponds to a horizontal arrow from a start point in the same row. Regardless of the presence of an edge token, each PE is always in one of two states: state 0 searching for a white to dark (0 to 1) transition or state 1 searching for the reverse transition. Thus the following eight cases may arise in each cycle.

* The contour tracing is included in the problem specification to allow other extended uses of the algorithm, for example, in case information along the contour (e.g. curvature) should be collected as part of the features to be accumulated at the end points.



Arrows show motion of edge tokens as data are pumped into the systolic array. Coincidence of some combination of token pairs yields an end point (marked as a square).

Fig. 5. Edge tokens during the extraction of start and end points in the pattern of Fig. 1.

A. PE_i *detects an (expected) transition*

A1. PE_i *does not contain an edge token and has not received one from* PE_{i-1}. This situation is exemplified by point A in Fig. 5 and corresponds to the case of detecting a start point. If the transition is a $0 \rightarrow 1$, two edge tokens corresponding to edge types e^1 (left) and e^2 (right) are generated. Else the edge tokens generated are for e^3 (left) and e^4 (right). The right edge token is forwarded to PE_{i+1} in the following cycle, while the left token is kept in PE_i until the opposite transition is detected later.

A2. PE_i *does not contain an edge token but it has received one from* PE_{i-1}. The detected transition indicates the continuation of the edge corresponding to the edge token just received. PE_i then forwards the edge token to PE_{i+1} to continue the trace. It also changes its state to detect a transition of the opposite type.

A3. PE_i *contains an edge token but has not received one from* PE_{i-1}. This is analogous to case A2. PE_i forwards its stored edge token to PE_{i+1} to continue the trace and changes its state to detect the opposite transition.

A4. PE_i *contains an edge token and has also received one from* PE_{i-1}. The situation is exemplified by pixel B in Fig. 5. The transition corresponds to the continuation of the edge for the edge token stored in PE_i. So PE_i should swap the two edge tokens and forward the old edge token it was holding to PE_{i+1} to continue the trace. Thus PE_i will continue the trace for the new edge token it has just received by switching its state.

B. PE_i *does not detect an (expected) transition*

B1. PE_i *contains an edge token and has also received one from* PE_{i-1}. This situation is exemplified by point C in Fig. 5. It corresponds to the scenario when two edge tracings first meet at a pixel which is not an edge pixel and an end point is thus formed. PE_i generates an end point token by including the needed information and discards both edge tokens afterwards.

B2. PE_i *does not contain an edge token but has received one from* PE_{i-1}. The edge corresponding to the received token has not been successfully extended in row i yet. So PE_i stores the received token to continue the trace in this row.

B3. PE_i *contains an edge token but has not received one from* PE_{i-1}. As in case B2, PE_i continues the trace without change.

B4. PE_i *does not contain an edge token and has not received one from* PE_{i-1}. This corresponds to the quiescent situation in which PE_i is looking for either a start point or a notification from PE_{i-1} to continue tracing.

The systolic algorithm can now be stated.

Systolic algorithm for contour tracing and extracting feature points

for 3M cycles *do*
  for all PE_i, $1 \leqslant i \leqslant M + 1$, *pardo*
    if the expected transition occurs *then*
      if PE_i receives an edge token from PE_{i-1} *then*
        if PE_i has a stored edge token *then*
          {Two contours are touching}
          send the stored edge token to PE_{i-1};
          store the received edge token
        else
          {No stored edge token in PE_i}
          {The transition corresponds to the continuation of the edge from PE_{i-1}}
          send the received edge token to PE_{i-1}

else
{No edge token from PE_{i-1}}
if PE_i contains a stored edge token *then*
send the stored edge token to PE_{i-1}
else
{No edge token in PE_i. The transition corresponds to a start point.}
extract the start point;
generate two edge tokens;
store the token corresponding to the left edge;
send the token corresponding to the right edge to PE_{i+1};
set the state to detect a transition of the opposite type
else {No transition}
if PE_i receives an edge token from PE_{i-1} *then*
if PE_i contains a stored edge token *then*
{This corresponds to an end point.}
extract the end point;
gather the required information from the two meeting edges;
kill the two edge tokens
else {No stored edge token in PE_i}
store the received edge token.

The feature points (and the corresponding edges) extracted by the systolic algorithm for the example are as marked in Fig. 5 (the skewed image) as well as Fig. 1(b) (the unskewed version). The correctness of the algorithm can next be considered.

Lemma 1. In general, PE_i contains at most one stored edge token and receives at most one edge token from PE_{i-1}.

Proof. This is immediate from the algorithm code. In particular, generation of token(s) occurs only when PE_i is in case A1 (start point detected) when it has neither stored nor received any token. So the above invariant is always preserved. ■

Lemma 2. The predecessor of end point x is the start point (among the two associated with x) for a $1 \rightarrow 0$ transition and the successor of x is the other with a $0 \rightarrow 1$ transition.

Proof. Consider a contour enclosing a white region. So the contour is traversed in the clockwise direction. Between two successive start points, the algorithm traces two edges to meet at an end point x. Since end point x has a higher rank than both start points, the predecessor of x must correspond to a $1 \rightarrow 0$ transition (edge type e^1 or e^4) and the successor of x must correspond to a $0 \rightarrow 1$ transition (edge type e^2 or e^3). Similar proof applies to a contour enclosing a dark region. ■

Theorem 1. The systolic algorithm correctly extracts the end point tokens after visiting all edge pixels.

Proof. First, any edge pixel in the image is always detected as it passes through the corresponding PE. In particular, if it is a start point at coordinate (i,j), PE_i will detect it from the skewing used, the absence of token from PE_{i-1} implies pixel at $(i-1,j-1)$ is not an edge pixel, and the absence of token in PE_i implies non-edge pixels at $(i-1,j)$ and $(i-1,j+1)$. Afterwards, the two edge traces (tokens) generated will be forwarded accordingly to continue the trace and lemma 1 ensures that the integrity of these traces are preserved. Eventually two such tokens collide at a non-edge pixel which satisfies the definition of an end point. From lemma 2, all relevant information is passed from an edge token to be used in generating an end point token. ■

From the space–time diagram of the skewed image shown in Fig. 5, the time complexity of the algorithm can be easily analyzed. While the image is sent in at a rate of M pixels per cycle, an entire image frame will percolate completely through the PE's in $3M$ cycles and an additional cycle is used to reset the PE's to ready themselves for the next frame. Since data (token) flow between two adjacent PE's is constant, the space complexity of each PE at worst grows with the size of an edge token ($O(\log M)$). Consequently, the size of a PE is small for all practical values of M. The end point tokens generated can be forwarded to stage 2 for further feature processing.

4. SYSTOLIC CHAIN COLLECTION

The feature extraction processor generates end point tokens from an image frame. These end point tokens appear in random ordering among the PE's in Section 3. Subsequently they have to be accumulated according to the order in which they appear in the corresponding contours for classification use. The problem is difficult because the ordering of two arbitrary end point tokens can be determined only if they are "adjacent" in the chain.

An abstract version of the problem in the form of ranking the elements in a two-way linked list systolically in optimal time is first posed, before the generalization to multiple contours (lists) is resolved. First the relation predecessor/successor is extended to end points tokens.

Definition. An end point token x is the predecessor/successor of another token y if the successor/predecessor (start point) of the end point for x is the same as the predecessor/successor (start point) of the end point for y. Thus, when two end point tokens meet, this predecessor/successor relation can be evaluated using the information contained in the tokens.

We can now imagine the end point tokens as elements of a two-way linked list with predecessor/successor relations. The abstract single linked list problem is stated as follows.

Input. A sequence of (randomised) elements $S_1,\ldots,S_n$ of a linked list.

Output. The ranks $R_1,\ldots,R_n$ of the elements

$S_1, \ldots, S_n$ respectively. (The rank of S_i is the number of elements preceding S_i in the list).

The difficulty in solving the above problem lies in the transitive relation (precedes) to be established between two arbitrary elements, and the fact that only systolic data flow and adjacent cell communication are allowed. For quite some time, it almost appeared that an optimal-time algorithm ($O(n)$) cannot be derived. The $O(n)$ complexity is justified by the fact that in the worst case an element must know the existence of all elements preceding it in the list and that a tree-network is not systolic.[13]

The design of an $O(n)$ systolic algorithm, however, is made possible with the following data structure. Imagine the n elements initially reside in n PE's, labelled PE_1 to PE_n. PE_i maintains a (maximum) sublist represented by the head, tail and count (size) of the sublist whose tail is S_i. The adjacent data communication allows PE_i to forward to PE_{i+1} (with wrap around from PE_n to PE_1) another (maximum) sublist abbreviated by D_{out} for PE_i and D_{in} for PE_{i+1} respectively. In every cycle, PE_i executes the following.

Systolic ranking algorithm to find the ranks of the elements in a doubly linked list

{initialization}
  for PE_i, $1 \leqslant i \leqslant n$, *pardo*
    S.head := predecessor of element in PE_i;
    S.tail := element stored in PE_i;
    S.count := number of elements in the partial list S;
    $D_{out} := S$
{computation}
  for n cycles *do*
    for PE_i, $1 \leqslant i \leqslant n$ *pardo*
      D_{out}.tail := D_{in}.tail;
      if S.tail = D_{in}.head *then*
        D_{out}.head := S.head;
        D_{out}.count := D_{in}.count + S.count -1
      else
        D_{out}.head := D_{in}.head;
      if S.head = D_{in}.tail *then*
        S.tail := D_{in}.head;
        S.count := S.count + D_{in}.count -1.

The ranking algorithm is illustrated in the space–time diagram shown in Fig. 6. Its correctness is next considered.

Lemma 3. At the beginning of cycle $j \leqslant n$, at PE_i (a) D_{in} is the maximum sublist (represented by head, tail and count) formed using elements $S_{i-1}, \ldots, S_{i-j}$ (with wrap-around, i.e. modulo-n) whose tail is S_{i-j}; (b) S is the maximum sublist formed using elements $S_i, \ldots, S_{i-j+1}$ whose tail is S_i.

Proof. Using induction on j, the lemma holds for $j = 1$. Assuming the lemma holds up to the k^{th} cycle, $k < n$, consider $j = k + 1$. According to the induction hypothesis and the algorithm code, in cycle k, PE_{i-1} merges two maximum sublists, say L_1 (corresponding to its D_{in}) and L_2 (corresponding to its S), to form two sublists L_3 (the new S) and L_4 (D_{out}). L_1 is the maximum sublist using elements S_{i-2} to S_{i-k-1} and with S_{i-k-1} as its tail. L_2 is that using elements S_{i-1} to S_{i-k} and with S_{i-1} as its tail. The two lists L_1 and L_2 involve a common set of elements except S_{i-k-1} and S_{i-1}. The concatenation code checks to see if S_{i-1} is predecessor of L_1. If so, L_4 is formed by the concatenation of L_1 and L_2 as the latter lists must be disjoint to satisfy the maximum property. If not, L_4 is simply equal to L_1. In both cases, L_4 is the maximum sublist involving elements $S_{i-1}, \ldots, S_{i-k-1}$ and with S_{i-k-1} as its tail. This is the needed D_{in} to PE_i at cycle $k + 1$. The proof of (b) involving S is

PE	Cycle 1		Cycle 2		Cycle 3		Cycle 4		Cycle 5		Cycle 6		Cycle 7		Cycle 8	
	S	D_{in}	S	D_{in}	S	D_{in}	S	D_{in}	S	D_{in}	S	D_{in}	S	D_{in}	S	D_{in}
1	a-a	g-h	a-a	f-g	a-a	d-e	a-a	d-f	a-a	a-b	a-a	c-d	a-a	a-c	a-a	a-a
a	1	2	1	2	1	2	1	3	1	2	1	2	1	3	1	1
2	b-c	a-a	b-c	g-h	b-c	f-g	b-c	d-e	b-c	d-f	b-c	a-b	a-c	c-d	a-c	a-c
c	2	2	2	2	2	2	2	2	2	3	2	2	3	2	3	3
3	c-d	b-c	b-d	a-a	b-d	g-h	b-d	f-g	b-d	d-e	b-d	d-f	b-d	a-b	a-d	a-d
d	2	1	3	1	3	2	3	2	3	2	3	3	3	2	4	4
4	a-b	c-d	a-b	b-c	a-b	a-a	a-b	g-h	a-b	f-g	a-b	b-e	a-b	b-f	a-b	a-b
b	2	2	2	2	2	1	2	2	2	2	2	4	2	5	2	2
5	e-f	a-b	e-f	c-d	e-f	a-c	e-f	a-a	e-f	g-h	e-f	f-g	e-f	a-e	a-f	a-f
f	2	2	2	2	2	3	2	1	2	2	2	2	2	5	6	6
6	d-e	e-f	d-e	a-b	d-e	c-d	c-e	a-c	a-e	a-a	a-e	g-h	a-e	e-g	a-e	a-e
e	2	2	2	2	2	2	3	3	5	1	5	2	5	3	5	5
7	f-g	d-e	f-g	d-f	d-g	a-b	d-g	c-d	c-g	a-c	a-g	a-a	a-g	g-h	a-g	a-g
g	2	2	2	3	4	2	4	2	5	3	7	1	7	2	7	7
8	g-h	f-g	f-h	d-e	f-h	d-f	d-h	a-b	d-h	c-d	c-h	a-c	a-h	a-a	a-h	a-h
h	2	2	3	2	3	3	5	2	5	2	6	3	8	1	8	8

Legend: $\overset{\text{x-y}}{\text{i}} \equiv$ x : head ; y : tail, i : count

Fig. 6. Space–time diagram illustration of ranking algorithm.

similar and is omitted. ■

The following theorem is a direct corollary of lemma 3.

Theorem 2. At the beginning of cycle n, both S and D_{in} at each PE_i has a count field equal to the rank of S_i.

Several minor modifications of the above algorithm have to be incorporated in adapting it to rank the end point tokens in handwritten numeral recognition. These arise because of the following circumstances.

P1. There are multiple end points per row in the image. For our application (through later experimental studies), we confine the maximum number of end points to three.

P2. The end point tokens in a contour actually form a circular list, not one with a unique head and tail.

P3. Multiple contours exist in an image frame.

To accommodate up to three end point tokens (P1) generated in each of the *PE*'s in the feature extractor of Section 3, a simple solution is proposed here. We associate three ranker *PE*'s with each extractor *PE* so that a distinct ranker *PE* will hold a distinct token, fed from the corresponding extractor *PE*. The ranker *PE* is triggered to perform ranking upon receipt of the systolic reset signal that marks the end of an image frame. Since there are $3M$ ranker *PE*'s (from row 2 to $M + 1$) and a column of M pixels is sent in every cycle, the ranker *PE*'s must operate at 3 times the clock rate of the extractor, or else the image frame input rate must be correspondingly slower (say one frame every $3M$ cycles). Observe that a ranker *PE* may be empty and thus contain a null list. During initializing, each ranker *PE* holding an end point will set its *S*.head to become the predecessor (start point). In case the *PE* is empty, *S*.head and *S*.tail are equal to null.

To accommodate P2 and P3, some nontrivial extension of the algorithm is required. Observing that the ranking algorithm ranks the tokens in a chain in every iteration of n cycles (with n *PE*'s), it should be modified to identify concurrently the unique head of the next chain to be ranked (if any) and preserve the integrity of all tokens yet to be ranked (not in the current list). This is accomplished by using the following theorem.

Theorem 3. If the elements in the ranker *PE*'s form one (or more) circular lists, at the end of an iteration (of n cycles), PE_i will hold *S*.head = *S*.tail.

Proof. This follows from the same reasoning used in the proof of lemma 3, as a complete list traversal is successful starting at any element of a list. The details are omitted. ■

Based on theorem 3, the chain collection array for our recognizer can be designed. It consists of a sequence of ranker, buffer and collector processors, one set for each distinct chain, as depicted in Fig. 7. A ranker processor consists of n ($n = 3M$) *PE*'s. They receive the end point tokens to be ranked in parallel, and execute the ranking algorithm. After n cycle, the ranked tokens are loaded in parallel into the n buffer *PE*'s. The buffer *PE*'s perform a circular shift of these tokens, except in case when a token contains *S*.head = *S*.tail and is leaving PE_1. This token will not be forwarded back to PE_n; instead, this token which belongs to the current list will be re-routed to PE_1 of the collector processor. A token received by the collector processor migrates down the collector *PE*'s and stops to reside in PE_j where j is the rank of the token. Thus at the end of n cycles, the buffer *PE*'s contain the tokens yet to be ranked while the collector *PE*'s hold the ordered list of tokens detected. The buffer *PE*'s now dump their contents to the next ranker processor, making room for the new data of the next image frame to be received.

The process of ranking, buffering and collecting tokens in a chain is pipelined so that the image input rate is not affected by this process. The total time to collect a chain is $2(3M)$ cycles after which the chain can be forwarded for classification. A distinct chain of an image is also detected and ordered every $3M$ cycles. Assuming at most c chains exist in an image frame, the area complexity of the *PE*'s is $O(cM\log M)$. The $\log M$ factor accounts for the size of a token, as in the case of the extractor processor. The concurrency of all processors is maximal as the processing of consecutive image frames are overlapped at these processors.

5. CLASSIFICATION

The use of chains of end point tokens in classification is highly application dependent. For our purpose of recognizing handwritten numerals, each chain represents a high order feature which will be called chain feature. To allow a high variability of writer style, we are concerned with detecting distinct features of a character, while ignoring irrelevant details. Obviously extensive experimentation should be involved, as will be outlined later.

The classification scheme we developed involves: (a) identification of holes (contours enclosing white regions), their sizes and relative positions, and (b) distinguishing significant cavities and their shapes in the outer contours (enclosing dark regions) as they appear in a certain sequence.

From the analysis in Section 2, a hole is identified when the lowest ranked start point in the contour shares a R_7 relation with another start point. The size of a hole is estimated by $(x_{max} - x_{min})$ $(y_{max} - y_{min})$ where x_{max}, x_{min}, y_{max}, y_{min} are the maximum and minimum values of the x and y coordinates of the feature points detected in the contour. Based on the size obtained, thresholds are set up to classify a hole as *B*(big), *M*(medium) or *S*(small). From these four parameters, it is also possible to estimate the centre

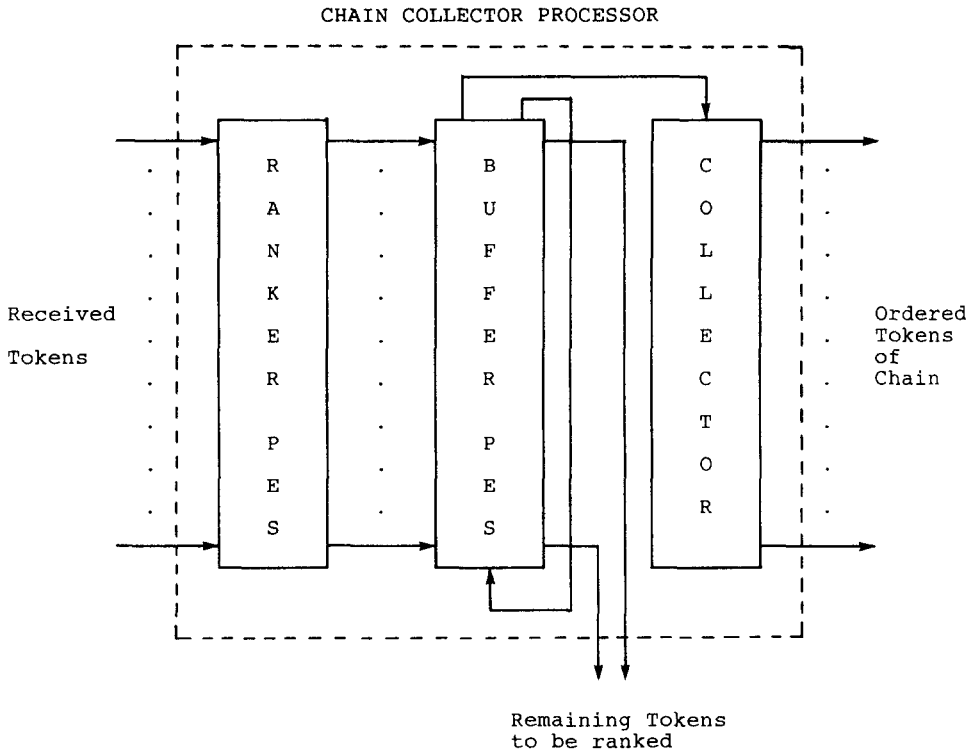


Fig. 7. Chain collector processor.

of the hole and classify the hole as L(left), R(right), U(up), D(down) or C(centre), as depicted in Fig. 8. Specifically, one can identify the centre of the hole by

$$(x_{\min} + \tfrac{1}{2}(x_{\max} - x_{\min}), y_{\min} + \tfrac{1}{2}(y_{\max} - y_{\min})).$$

Thus different hole features can be detected, for example, *BLH* (big left hole), *MUH* (medium upper hole) and so on.

The hole classifier designed is integrated into PE_1 of the collector processor described in Section 4. As the end point tokens of a chain are fed serially into PE_1 of the collector processor, PE_1 performs the above additional task of sampling all tokens, detecting the critical R_7 relation, the values of $x_{\max}$, $x_{\min}$, $y_{\max}$, $y_{\min}$, and computing the size and location of the hole in case a hole does exist. As all tokens pass through PE_1, it is obvious the above is computable, and after the passage, the detected hole and its class are available as a distinct hole feature (such as *BLH*). In case the critical R_7 relation does not exist, however, the chain is not a hole and must be subject to further classification according to (b) above.

A chain of tokens corresponding to an outer contour (enclosing a dark region) are classified explicitly. In order to capture cavity shapes, a relation R_i is further distinguished into three types. (1) $R_{i,1}$: in case the predecessor start point has a rank much lower than the successor. (2) $R_{i,2}$: in case both the predecessor and successor start points have approximately equal ranks. (3) $R_{i,3}$: in case of the reverse situation in $R_{i,1}$. An example is depicted in Fig. 9. In

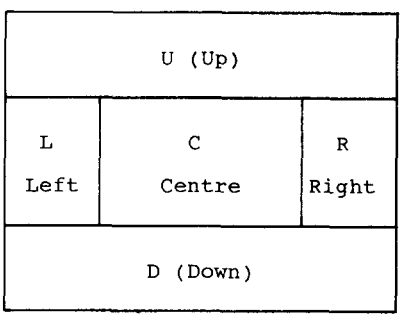


Fig. 8. Hole position classification.

the following, we assume an end point token already carries this information which is obtainable by comparing the ranks of the predecessor and successor start points of the end point.

The classification of a chain residing in a collector processor can be performed using a PLA-type array classifier, as depicted in Fig. 10. Column i in the array is responsible for recognizing a specific chain sequence. For example, if the chain sequence is ($R_{1,2}$; $R_{4,2}$), then PE_1 in this column will send a match output to PE_2 in this column if its horizontal token input contains $R_{1,2}$. In turn (the following cycle), PE_2 will forward a match output downward if it receives a match from PE_1 and its horizontal token input is $R_{4,2}$. The subsequent PE's down the column will simply forward the match as they are not programmed to check for further feature relations. Thus the tokens

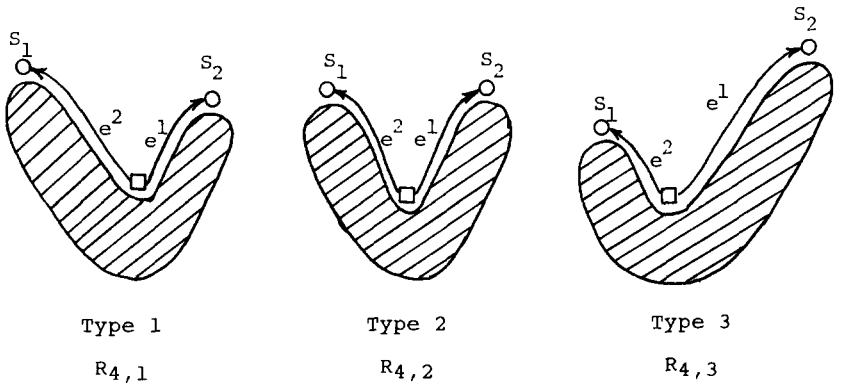


Fig. 9. Further classification of relations to distinguish cavities.

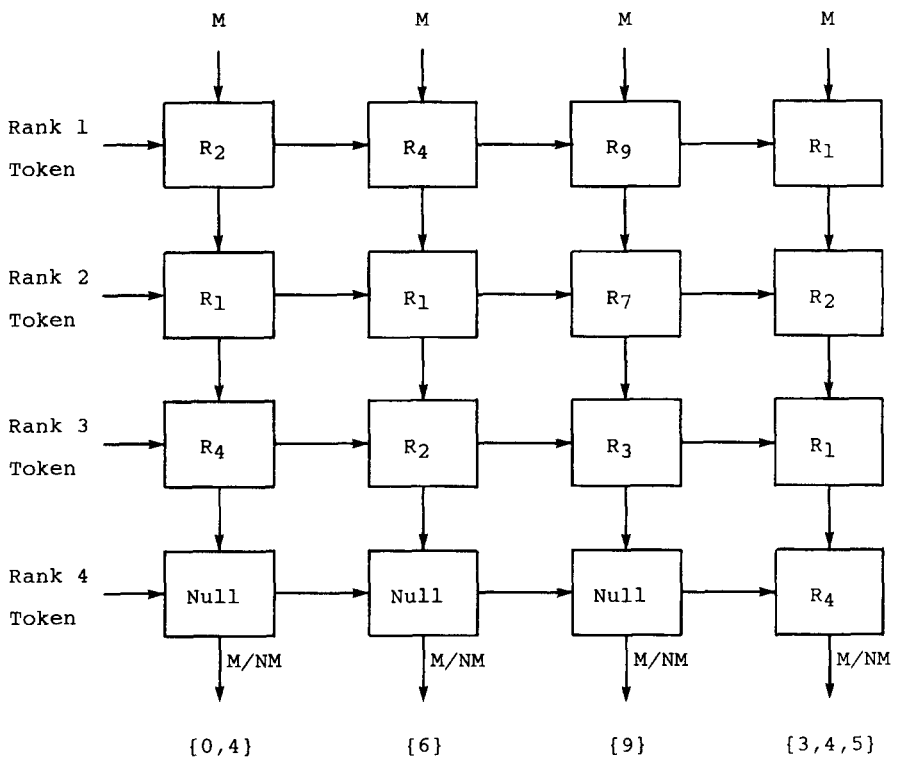


Fig. 10. Classifier example.

march systolically horizontally while the match (or no match) marches similarly downward. The size of this array is $n \times k$ where k is the number of distinct chain features in the classification.

A simulation experiment has been carried out for the whole design. The test data consisted of 1200 handwritten numerals (120 numerals per digit) obtained from U.S. postal codes. During this experiment, the chain classes are mapped into the corresponding character classes. For example, the classification table obtained using horizontal scanning is shown in Table 1. The following observations also result from this experiment.

(1) Most of the numerals can be classified using the chain classes extracted using a horizontal and a vertical scan. To differentiate between some confusing numerals involving 1 and 7, or 4 and 9, an additional diagonal scan is necessary.

(2) The classification scheme is applicable when the handwritten numerals are rotated by an angle between -15 and $+15$ degrees.

The classification table for horizontal scan contains

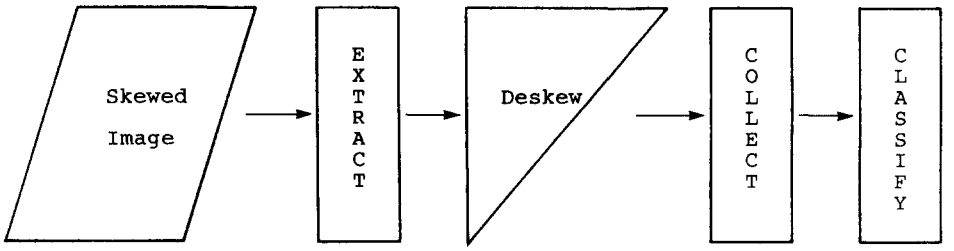


Fig. 11. Overall system organization.

Table 1. Classes of features from horizontal scan

Class number	Horizontal	Characters in class
1	$(R_{12,2})$, *HBL*, *HBR*	$\emptyset$, 8
2	$(R_{12,2})$, *HB*	$\emptyset$
3	$(R_{12,2})$, *HBU*	9
4	$(R_{12,2})$, *HML*, *HBR*	$\emptyset$
5	$(R_{12,2})$, *HMU*	4, 9
6	$(R_{12,2})$	1, 2, 3, 5, 7
7	$(R_{12,2})$, *HSD*	2
8	$(R_{2,2})$, $(R_{1,2}; R_{4,2})$, *HBL*, *HBR*	$\emptyset$
9	$(R_{2,3}; R_{1,2}; R_{4,3})$, *HBL*, *HBR*	$\emptyset$
10	$(R_{2,2}; R_{1,2}; R_{4,2})$	$\emptyset$, 4
11	$(R_{2,2}; R_{1,2}; R_{4,3})$, *HMU*	9
12	$(R_{1,2}; R_{2,3}; R_{1,2}; R_{4,3})$	3, 4, 5
13	$(R_{2,3}; R_{1,2}; R_{6,3}; R_{7,2}; R_{10,1}; R_{7,2}; R_{3,1})$	3
14	$(R_{12,3})$, *HBU*, *HBD*	8
15	$(R_{12,3})$, *HBU*, *HMD*	8
16	$(R_{12,3})$, *HMU*, *HMD*	8
17	$(R_{2,3}; R_{1,2}; R_{6,3}; R_{7,2}; R_{3,1})$	0, 3, 5
18	$(R_{4,1}; R_{1,2}; R_{2,1})$, *HBD*	6
19	$(R_{4,1}; R_{1,2}; R_{2,1})$, *HMD*	6, 8
20	$(R_{4,1}; R_{1,2}; R_{2,1})$	6, 4, 9
21	$(R_{4,1}; R_{1,2}; R_{9,3}; R_{7,2}; R_{3,1})$	5
22	$(R_{4,1}; R_{1,2}; R_{2,1})$, *HMU*, *HMD*	8
23	$(R_{4,1}; R_{7,2}; R_{3,1}; R_{1,2}; R_{2,1})$	6
24	$(R_{9,3}; R_{7,2}; R_{10,1}; R_{7,2}; R_{3,1})$, *HMD*	2
25	$(R_{9,3}; R_{7,2}; R_{10,1}; R_{7,2}; R_{3,1})$	2
26	$(R_{9,3}; R_{7,2}; R_{3,1})$ *HMU*	9
27	$(R_{9,3}; R_{7,2}; R_{3,1})$, *HMU*, *HMD*	8
28	$(R_{9,3}; R_{7,2}; R_{3,1})$	0, 1, 3, 5, 7

29 character classes (as shown in Table 1*) and that for vertical scan contains 35 character classes.

6. CONCLUSION

We have proposed and experimented with a set of systolic algorithms applicable in handwritten character recognition. The salient features of these algorithms include simple processing cell (computation), locality of communication and maximal pipelining. Feature extractors involving a contour tracer and a linked-list ranker are novel designs which are excellent candidates for direct VLSI implementation. The availability of VLSI and its future successors should gradually transfer part, if not all, of a pattern recognition system into a more hardware-based realization. Local and global feature extractions are promising candidates. Complex recognition problems, such as handwritten character recognition, can become a successful reality when both hardware and recognition methodology have progressed far enough to be able to handle the wide spectrum of variability of image classes in a space and time efficient manner. The parallel (systolic) algorithms presented here have optimal area and time complexities (under systolic constraints). However, their efficient realizations in VLSI should be studied when actual area and time cost (rather than complexities) are of concern in final system design.

* The classification table is generated after an additional smoothing phase is done on the outer-chain to remove short edge features: an edge with vertical height shorter than three pixels is merged with its neighbour by concatenating two end point tokens into one. This can be done at the collector processor, but is omitted in the description here for brevity.

REFERENCES

1. H. T. Kung and R. L. Picard, One-dimensional systolic array for multi-dimensional convolution and resampling, *VLSI for Pattern Recognition and Image Processing*, K. S. Fu, Ed., Vol. 13, *Springer Series in Information Science* (1984).
2. Hsi Ho Liu and K. S. Fu, VLSI Arrays for minimum-distance classifications, *VLSI for Pattern Recognition and Image Processing*, K. S. Fu, Ed., Vol. 13, *Springer Series in Information Science* (1984).
3. H. D. Cheng and K. S. Fu, Algorithm partition and parallel recognition of general context-free languages using fixed-size VLSI architecture, *Pattern Recognition* **19**, 361–372 (1986).
4. H. T. Kung, Why systolic architectures, *IEEE Comput.* 37–46 (1982).
5. H. T. Kung and C. E. Leiserson, Systolic arrays (for-VLSI), *Sparse Matrix Symp. SIAM*, pp. 256–282 (1978).
6. T. E. Calvert, Nonorthogonal projections for feature extraction in pattern recognition, *IEEE Trans. Comput.* **C-19**, 447–452 (1970).
7. B. V. Dasarathy and K. P. B. Kumar, Chitra: cognitive handprinted input-trained recursively analyzing system for recognition of alphanumeric characters, *Int. J. Comput. Inf. Sci.* **7**, 253–282 (1978).
8. P. Ahmed and C. Y. Suen, Edge classification and extraction of shape features, *7th Int. Conf. Pattern Recognition*, pp. 593–596 (1984).
9. M. T. Y. Lai and C. Y. Suen, Automatic recognition of characters by Fourier descriptors and boundary line encoding, *Pattern Recognition* **14**, 383–393 (1981).
10. M. Shridhar and A. Adrelin, High accuracy character recognition algorithms using Fourier and topological descriptors, *Pattern Recognition* **17**, 515–524 (1984).
11. P. Ahmed, Computer recognition of totally uncon-

strained handwritten ZIP codes, Ph.D. Thesis, Dept of Computer Science, Concordia University, Montreal, Canada (1986).
12. A. C. Shaur, A formal description scheme as a basis for picture processing system, *Inf. Control* **14**, 9–52 (1969).
13. H. F. Li and R. Jayakumar, Systolicity: a notion and characterization, *J. Parallel Distributed Comput.* **3**, 373–397 (1986).

About the Author—HON F. LI received the B.Sc. degree with highest honours from the University of California, Berkeley in 1972, and remained there until 1975 when he completed a Ph.D. degree in the area of parallel-pipelined computer architecture. Subsequently he joined the University of Illinois, Urbana-Champaign, as an Assistant Professor in the Computer Engineering Group, Department of Electrical Engineering. He went to Hong Kong in 1977 and served as Lecturer and later, Senior Lecturer, in the Department of Electrical Engineering, University of Hong Kong, where he developed both an undergraduate and a graduate program in computer engineering. In 1984 he joined Concordia University, Montreal, where he is now a Professor of Computer Science. His research interests cover the general areas of parallel and distributed processing, algorithms and architectures, and system specification.

About the Author—R. JAYAKUMAR received the B.E. (Honours) degree in Electronics and Communication Engineering from the University of Madras, Madras, India in 1977, the M.S. degree in Computer Science from the Indian Institute of Technology, Madras, India in 1980, and the Ph.D. degree in Engineering and Computer Science from Concordia University, Montreal, Canada in 1984. Since 1984, he is an Assistant Professor of Computer Science at Concordia University, Montreal, Canada. His research interests are in VLSI algorithms and architectures, fault-tolerant VLSI systems, VLSI design automation, and graph theory and graph algorithms.

About the Author—MELAD YOUSSEF GHABRIAL received the B.Sc. degree in Electrical Engineering from Ain Shams University, Cairo, Egypt in 1982, and he is currently fulfilling the requirements for the Master of Computer Science degree at Concordia University, Montreal, Canada. From 1982 to 1986 he worked as a software field engineer for T.E.A. Computers, Cairo, Egypt and he is currently working as a technical support engineer with Planmatics Inc., Montreal, Canada. His research interests include design and analysis of parallel algorithms for pattern recognition and machine intelligence.