REGIONAL PROJECTION TRANSFORMATION AND ITS SYSTOLIC IMPLEMENTATIONS

XIANG CHENG

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

JUNE 1994

ISBN 0-315-97624-1

Canada

Abstract

Regional Projection Transformation and Its Systolic Implementations

Xiang Cheng

One difficulty in making pattern recognition systems practically feasible, and hence more popularly used, is the requirement of computation time. Most pattern recognition algorithms have been regarded computationally expensive. Parallel processing has been used as an effective way to solve this problem. With the advance of VLSI technology, it is possible to put thousands of electronic components on one silicon chip. This not only reduces the cost of processors, and increases the communication speed, but also makes immense impact on the algorithm design. It requires algorithm designer to take full advantage of pipelining and parallelism provided by VLSI architecture in algorithm design in order to reduce computation time. Such VLSI oriented algorithm design has been applied to a wide range of areas including pattern recognition. Numerous parallel algorithms and VLSI architectures have been proposed to solve various problems in pattern recognition.

Feature extraction plays an important role in pattern recognition system design. In order to improve the system performance, various feature extraction approaches have been proposed and widely used in pattern recognition. However, these approaches still encounter great difficulty in dealing with multicontour patterns including compound patterns and patterns with holes in them.

In this thesis, a new approach called *regional projection transformation* (RPT) is proposed to solve these kind of problems. This approach simplifies the recognition of an image by transforming it into an integral object. It is very useful in processing patterns containing unconnected patterns, patterns with isolated noises or patterns with internal contours.

To further reduce the computation time, a parallel regional projection transformation is presented. Its implementations on different systolic architectures are also studied. The proposed parallel regional projection transformation has time complexity of $O(N)$ compared with $O(N^2)$ of the sequential RPT approach.

Experiments show that quality of features extracted by RPT is better than other approaches in processing multicontour patterns. Encouraging results have also been achieved when this approach is used to recognize Chinese characters mixed with English characters and numerals.

Acknowledgements

I am grateful to my thesis advisor Dr. Lixin Tao for his support and advice during my study at Concordia University. I would also like to express my gratitude to the members of my thesis defense committee Dr. Stan Klasa, Dr. Tao Li, and Dr. Ching Y. Suen for their comments and suggestions for my research.

I would like to especially thank Dr. YuanYan Tang, for giving me every help during my research work for the thesis.

More than anyone else, I would like to thank my grandmother, my parents. It was their continued support and encouragement that made this degree possible.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In most applications of recognition technology, the dimensionality of feature space tends to be very large. The number of dimensions may be up to several hundreds. This great dimensionality is a major cause of the cost and practical limitations of recognition technology. Consequently, the selection of a stable and representative set of features, and the reduction of the dimensionality of the feature space play an important role in recognition system design. It not only affects the system recognition rate, but also the system efficiency. Feature extraction is one of the techniques to overcome these problems. Feature extraction reduces the dimensionality of feature space by concentrating the information in the image into a few highly selective features. It finds wide applications in pattern recognition, computer vision, image processing and other areas.

According to the manner in which features are derived, feature extraction methods can be classified into three categories: local feature method, global feature method and structure feature method. Local feature method uses the limited extent of the region to compute each feature. Although this method is a general one and is equally applicable to both character recognition and general shape recognition, its ultimate accuracy is expected to be limited. Structure methods assemble a model of a pattern based on the structure which exists in the pattern. By examining the way how the pattern is structured, the pattern is recognized. Although this approach is effective for some kinds of patterns (like Chinese characters) which contain rich structure

information, it is very difficult to be extended to general applications. The global feature method employs a global transformation to summarize the information in the input image. An ordered subset of the transform coefficients is taken to be the feature vector. By considering the image as a whole, it treats the input more abstractly than local feature method. It is also more tolerant of local variations. As a general method, it can be used in recognizing any kind of patterns.

Projection methods are a subclass of global transformations. The main characteristic of these methods is the compression of the data through a projection. Black pixel counts are taken along parallel lines through the image area to generate marginal distributions. The distribution may be of local features as well as simple pixel densities. One dimensional transforms can also be applied to further summarize the projection. Projection methods have been widely used in Chinese character recognition as well as shape recognition [135, 46, 49, 90, 48, 2].

However, when these methods are used to process following patterns:

(1) compound patterns which contain unconnected subpatterns;
(2) patterns with isolated noise;
(3) patterns including internal contours.

it will still encounter great difficulty due to (1) more contours to deal with; (2) relations between isolated parts have to be carefully established; (3) more complicated feature extraction; (4) more time required for processing, etc.

In this thesis, a new approach called regional projection transformation (RPT) is proposed to solve these kind of problems. RPT is a transform based on regional projections which concentrate the compound pattern into an integral object with only one outer contour and no internal contours. The basic principle of the RPT is that all pixels of a pattern are projected onto some base lines termed projection bases such as horizontal, vertical, diagonal, etc, and a contour chain is extracted from projected integral pattern for further processing.

Since the advance of VLSI technology, it is possible to put many hundreds of thousands of gates to be placed on a single silicon chip. This not only reduces the cost for processors and increases the communication speed, but also makes an immense

impact on the computer architecture, and algorithm design. With VLSI technology, we can build special VLSI chips by implementing algorithms directly in hardware. Such specially designed VLSI chips are usually attached to host machines. They solve problems more efficiently by taking advantage of pipelining and parallelism in algorithms. In the last decade, such VLSI oriented algorithm design has attracted a lot of attention. It has been applied to a wide range of areas including pattern recognition and image processing.

The essential parallelism in proposed RPT facilities its implementation using VLSI architecture. In this thesis, we also propose two parallel RPT approaches and discuss their implementations using different systolic architectures. One is based two dimensional mesh architecture, the other uses a linear processor array. Both of parallel RPT approaches have time complexity of $O(N)$ comparing with $O(N^2)$ when uniprocessor is used.

We also conduct experiment to evaluating the features extracted by the RPTs. We applied this approach to recognizing a large set of characters which contain many compound patterns. Encouraging results have been obtained and they will also be reported in this thesis.

This thesis is organized as follows:

In Chapter 2, we first give a concise literature survey on VLSI oriented algorithm design, and its application to pattern recognition and image processing. We also give a review of various feature extraction methods that have been proposed.

Chapter 3 introduces two regional projection transformations, the DDRPT and HVRPT. First, a theoretical analysis of the compound and integral patterns are discussed. Then, DDRPT and HVRPT are described and analyzed. The pattern transformed by these two transformations possesses several important properties which can simplify contour processing. These properties are presented and proofs of two theorems are also given in this Chapter.

In Chapter 4, we first discuss the parallelism existing in both DDRPT and HVRPT. The essential parallelism in DDRPT and HVRPT facilities its implementation using systolic architecture. Compared with $O(N^2)$ time complexity in sequential approach,

both parallel DDRPT and HVRPT have the time complexity of $O(N)$. In this chapter, a parallel contour processing algorithm with constant time complexity and its systolic implementation will also be described.

Chapter 5 presents a combinatory parallel approach termed regional projection contour transformation (RPCT) as a further improvement of the parallel RPT proposed in Chapter 4. The advantage of RPCT over RPT is that using RPCT, regional projection and contour processing can be done concurrently, yielding a high degree of parallelism. A systolic architecture using a linear array has been designed based on the *canonical mapping methodology* described in [75]. When $N/2$ element vector is used to process a pattern with the size of $N \times N$, it can speed up the recognition process considerably by reducing the complexity to $O(N)$ from $O(N^2)$ in uniprocessor case.

In Chapter 6, we will show the results of evaluating the features extracted by the RPTs. An important theorem will be introduced to estimate the quality of features extracted. This theorem is used to evaluate the DDRPT and HVRPT comparing with other approaches, such as, cell histogram(CH), crossing counts and shading(CCS), and elastic partitioning (EP).

Chapter 7 concludes the thesis with a summary of this research and the discussions about the future research work.

Chapter 2

Literature Survey

2.1 VLSI Systems

Very-large-scale integration (VLSI) technology makes a great progress in development of fast operation and low power consumption switching elements and vastly increase the density of the circuit. It has promised the availability of building millions of switching elements on a single silicon chip. In addition, high density also assures high performance and high reliability [85, 64]. As mentioned by Mead [85], VLSI electronics presents a challenge not only to those involved in the development of fabrication technology, but also to computer scientists and computer architects. The ways in which digital systems are structured, the procedures used to design them, the trade-offs between hardware and software, and the design of computational algorithms will all be greatly affected by the coming changes in integrated electronics.

The impact of VLSI on the computer architecture and computer technology has been discussed in several articles [101, 31, 102, 11, 118]. It has created a new architecture horizon in implementing parallel algorithms directly on hardware. Special-purpose VLSI chips can function as peripheral devices attached to a conventional host machine.

The algorithm suitable to be implemented by systolic architectures should have the following properties [85, 64, 36]:

- It only require a few different types of processing elements.

- The data and control flow have to be simple and regular.
- The algorithms should have the ability to perform the operations with high degree of space and time concurrency by using extensive pipelining and multi-processing techniques.

Kung has done some pioneer work with the concept of systolic algorithm [69, 61, 65]. The definition and the detail description of systolic algorithm can be found in [65, 69, 78].

Several basic geometry structures are suitable to make the systolic implementation of algorithms, they are as follows [61] :

1. One-dimensional linear arrays (Figure 1(a)): This is the simplest way of connecting processors and corresponds to the classical concept of pipeline computation.
2. Square arrays (Figure 1(b)) : It is natural for problems involving matrices, such as graph problem defined in terms of adjacency matrices, numerical solutions to discretized partial differential equations.
3. Hexagonal arrays (Figure 1(c)) : It enjoys the property of symmetry in three directions and eliminates a possible separate loading or unloading phase for processors, hence it can substantially reduce the complexity of processing elements.
4. Trees (Figure 1(d)) : It supports logarithmic-time broadcast, search, or fan-in, which is theoretically optimal. The drawback is that the processors at high levels of the tree may become bottle-necks of the majority if communications are not confined to processors at low levels.
5. Shuffle-exchange networks (Figure 1(e)) : Assume that a network has $n = 2^m$ nodes, where m is an integer and that the nodes are named $0, 1, 2, \cdots, 2^m - 1$. Let $i_m i_{m-1} \cdots i_1$ denote the binary representation of any integer $i, 0 \leq i \leq 2^m - 1$, the shuffle function is defined by:

$$S(i_m i_{m-1} \cdots i_1) = i_{m-1} \cdots i_1 i_m$$

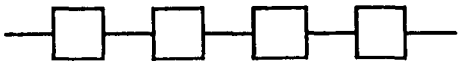
and the exchange function is defined by :

$$E(i_m \cdots i_2 i_1) = i_m \cdots i_2 \bar{i}_1$$

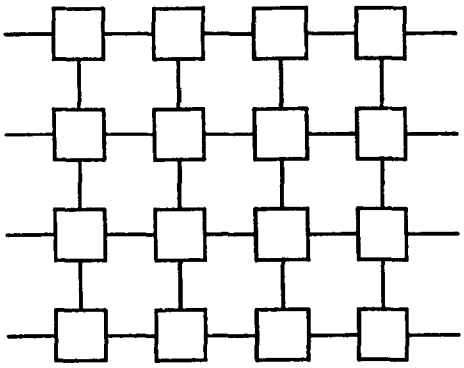
The network is called a shuffle-exchange network if node i is connected to node $S(i)$ for all i, and to node $E(i)$ for all even i. By using the exchange and shuffle connections alternately, data originally at two nodes whose names differ by 2^i can be brought together for all $i = 0, 1, \cdots, m-1$. This type of communication structure is common to a number of algorithms. It is shown that the bitonic sort of n elements could be carried out in $O(logn)$ steps on the shuffle-exchange network when the processing elements are capable of performing comparison exchange operations. But it has a very low degree of regularity and modularity, and has wires of various lengths, this can be a serious drawback for VLSI implementation.

Table 1 gives the examples of systolic algorithms [78]. Table 2 gives a partial list of problems for which systolic solutions exist [62].

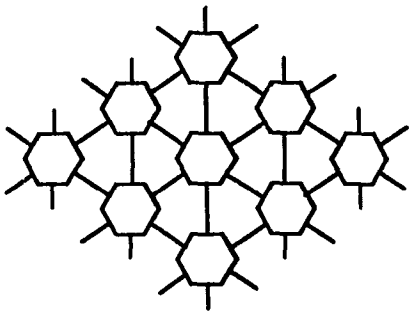
There are many articles discuss the systolic implementations for particular applications. The systolic implementation of signal and image processing algorithms have been discussed in [78]. [142] describes a liner systolic array capable of evaluating a large class of inner-product functions used in signal and image processing. These include matrix multiplication, multidimensional convolutions using fixed or time-varying kernels, as well as various nonlinear functions of vectors. [62] gives the new designs of special-purpose devices for filtering, correlation, and discrete Fourier transform. A novel systolic algorithm for performing the two-dimensional convolution operation is introduced in [61]. A chip performing the two dimensional convolution in signal and image processing is described in [63]. Denoting by u, the cycle time of the basic cell, the chip allows convolving a $k \times k$ window with an $n \times n$ image in $O(n^2u/k)$ time, using a total of k^3 basic cells. [73] describes a systolic array for the computation of n-dimensional convolution for any positive integer n. The systolic array utilizes a second level of pipelining by allowing the processing elements themselves to be pipelined to an arbitrary degree.



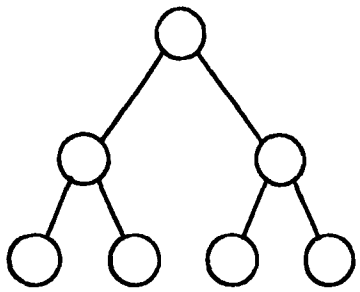
(a). One-dimensional linear array



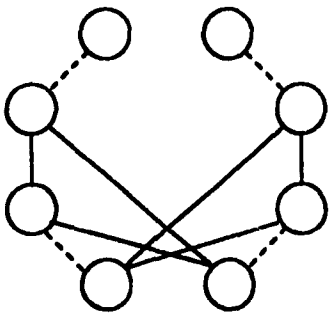
(b). Square array



(c). Hexagonal array



(d). Tree



(e). Shuffle-exchange network

Figure 1: Computational structures.

Table 1: Partial list of problems having systolic solutions

COMMUNICATION GEOMETRY	EXAMPLES
1-DIM linear arrays	Matrix-vector multiplication FIR Filter Convolution DFT Carry pipelining Pipeline arithmetic units Real-time recurrence evaluation Solution of triangular linear systems Constant-time priority queue, on-line sort Cartesian product Odd-even transposition sort
2-DIM square arrays	Dynamic programming for optimal parenthesization Numerical relaxation for PDE Merge sort FFT Graph algorithms using adjacency Matrices
2-DIM hexagonal arrays	Matrix multiplication Transitive closure LU-Decomposition by Gaussian elimination without pivoting
Trees	Searching algorithms Queries on nearest neighbor, rank, etc. NP-Complete problems Systolic search tree Parallel Function Evaluation
Shuffle-exchange networks	FFT Bitonic sort

Table 2: Examples of systolic algorithms

Signal and image processing	FIR and IIR filtering and 1-D convolution 2-D convolution and correlation discrete Fourier transform interpolation 1-D and 2-D median filtering geometric warping
Matrix arithmetic	matrix-vector multiplication matrix-matrix multiplication matrix triangularization (solution of linear systems, matrix inversion) solution of triangular linear systems solution of Toeplitz linear systems QR-decomposition (least squares computations, covariance matrix inversion) singular value decomposition eigenvalue problems.
Non-numeric applications	data structure–stack and queue, priority queue, searching, and sorting. graph algorithms–transitive closure, minimum spanning trees, and connected components geometric algorithms–convex hull generation language recognition–string matching and regular expressing dynamic programming polynomial manipulation–multiplication, division, and greatest common divisor integer greatest common divisors relational data-base operation

One-dimensional systolic array for performing two or higher dimensional convolution and resampling is presented in [72]. [74] describes the development of a wavefront based language and architecture for a programmable special-purpose multiprocessor array. The hardware and the languages lead to a programmable wavefront array processor (WAP). The WAP blends the advantages of the dedicated systolic array and the general purpose data-flow machine. Two VLSI structures for the computation of the discrete Fourier transform are presented in [7]. A network which is able to compute, in parallel, the FFT's of arbitrary partitions in power of two of the N input elements is described in [6]. [117] surveys nine designs for VLSI circuits that compute N-element Fourier transforms. The designs exhibit an area-time tradeoff. The problem of finding a greatest common divisor (GCD) of any two nonzero polynomials is fundamental to algebraic and symbolic computations, as well as to the decoder implementation for a variety of error-correcting codes. The efficient VLSI solutions to both the GCD problem and the extended GCD problem are discussed in [10]. [63] reviews issues in the design of special-purpose VLSI chips in general, and suggests VLSI designs for polynomial multiplication and division, which are basic functional modules in algebraic computation. A new class of partitioned matrix algorithms is developed in [51]. The following four matrix computations are shown systematically partitionable into submatrix operations, which are feasible for direct VLSI implementation.

- L-U decomposition of a dense matrix
- Inversion of a triangular matrix
- Multiplication of two compatible matrices
- Solving a triangular system of equations.

[26] presents a method for the design of problem-size independent VLSI systolic array with limited I/O bandwidth. VLSI algorithms partition problems have been investigated in [37, 20]. [89] proposes a solution for mapping an arbitrary large QR algorithm into smaller VLSI array processors. [99] describes a matrix multiplication

algorithm on a linear array of processing elements. VLSI array design under constraint of limited I/O bandwidth of the host system or the computing array is discussed in [81]. The area-time complexity of sorting is analyzed under an updated mode of VLSI computation. Using the new mode, [117] briefly describes thirteen different designs for VLSI sorters. A class of designs of a new interconnection network, the proposed cube-connected cycles (PCCC), which can implement stable bitonic sorting is given in [5]. A new VLSI architecture which allows many problems to be solved quite efficiently on chips with very small processing areas is proposed in [54]. The sorting problem is discussed in more detail. The VLSI chip which consists of a mesh of trees is proposed in [8]. Two classical algorithms, i.e., merge sort and bitonic sort, are modified to efficiently solve the external sorting problem using this chip. The VLSI implementation of a Reed-Solomon encoder using Berlekamp's bit-serial multiplier algorithm is described in [50]. The VLSI implementation of Reed-Solomon decoder is proposed in [59]. A family of VLSI circuits is presented to perform open convolution, i.e., polynomial multiplication [4]. A combinational limit to the computing power of VLSI circuits is discussed in [124]. In [131], several methods for computing the FFT in hardware are reviewed. Pipeline structures for Cooley-Tukey algorithm and the Good prime factor algorithm are presented. Two interconnection networks for parallel processing, namely the orthogonal trees network and the orthogonal tree cycles are described in [91]. An improved min-cut algorithm for partitioning VLSI network is described in [58]. [129] describes asynchronous and clocked control structures for VLSI based interconnection networks. In [96], a VLSI network for the multiplication of two N-bit integers, for vary large N. The network, which is based on the discrete Fourier transform, has an extremely regular mesh structure, and thus all wires have approximately the same length. [9] shows that addition of n-bit binary member can be performed on a chip with a regular layout in time proportional to $logn$ and with area proportional to n. [114] describes a residue multiplier having a $48 - 73$ bit dynamic range, capable of performing 10 M multiplications. [141] presents two systolic architectures for performing the product-sum computation $AB + C$ in the finite field $GF(2^m)$ of 2^m elements, where A, B, and C are arbitrary elements of $GF(2^m)$. The application of VLSI to database has been described in [68, 110]. Systolic priority

queues problem is described in [77]. [94] describes a dictionary machine that is suitable for VLSI implementation. The machine supports the operations of SEARCH, INSERT, DELETE, and EXTRACTION on an arbitrary ordered set. A binary tree machine which can handle all the dictionary machine and priority queue operations as well as some other data queries is given in [109]. [56] shows that a machine in which the processors are interconnected as a binary tree can support all the dictionary and priority queue operations as well as some other data queries. The PSC, a programmable systolic chip, is a high performance, special-purpose, microprocessor intended to be used in groups of tens or hundreds for the efficient implementation of a broad variety of systolic arrays is described in [35, 34]. In [67], two important issues in systolic array designs are addressed: How is fault tolerance provided in systolic arrays to enhance the yield of wafer-scale integration implementation? And, how are efficient systolic arrays with two levels of pipelining designed? [71, 66] describe 32-bit floating-point systolic array processor which was built at CMU. Using off-the-self integrated circuits. It can operate at a rate of 100MFLOPS and perform various transformations. [15] proposes an array structure which is programmable by a host machine and uses control buffers to broaden the scope of executable algorithms. In [143], orthogonally-connected, reconfigurable square arrays are proposed for pattern analysis and image processing. Highly parallel VLSI computing structures consist of many processing elements operating simultaneously. In order for such processing elements to communicate among themselves, some provision must be made for synchronization of data transfer. [33] provides a spectrum of synchronization models; based on the assumptions made for each model, theoretical lower bounds on clock skew are derived, and appropriate of best-possible synchronization schemes for large processor arrays are proposed. The area-efficient layouts problems are discussed in [78]. Area-time complexity of the discrete Fourier transform is studied with respect to a new model of computation appropriate to VLSI technology [115]. Optimal layouts for the shuffle-exchange graph and other networks are presented in [76]. Area-time tradeoff for matrix multiplication and related problems in VLSI models have been shown in [105]. [1] shows that the communication considerations alone dictate that any VLSI design for computing the $2n$-bit product of two n-bit integers must satisfy

the constraint $AT^2 \geq \frac{n^2}{64}$, where A is the area of the chip and T is the time required to perform the computation. Area-time bounds on VLSI circuits for context-free language recognition, for the evaluation of propositional calculus formula and for set quality and disjointness questions are described in [9]. A class of matrix arithmetic networks is proposed [53] for implementing the feature extraction algorithm and for generating linear discriminant vectors in pattern classification. The new algorithms for two important combinatorial problems, dynamic programming and transitive closure, are proposed in [47]. These algorithms are suitable for direct implementation in VLSI. The recognition of regular languages with programmable building-block is described in [39]. A systolic array for the Minimum-Distance Classification (MDC) which is capable of computing L_1, L_2 and L_∞ metrics is presented in [80]. A systolic array for the MDC of string is also described. Four VLSI designs for a line and curve detection chip are presented in [27]. Each method is based on Ontanari's dynamic programming algorithm. The VLSI implementation of hierarchical scene matching is described in [22]. [3] describes one- and two-dimensional pattern matching oriented systolic array processors that support, respectively, the detection of all repetitions in a string x and the statistics of all substrings of x with and without overlap. The linear classifier algorithm which can be computed using systolic arrays at the word and bit levels is described in [123]. [25] presents two VLSI architectures for the recognition of context-free languages and finite-state languages. The VLSI implementation of Earley's algorithm is described in [24]. The error-correcting recognition algorithm has also been simulated on a triangular VLSI array. Two algorithms which can recognize general context-free languages without restriction on the length of the input strings are proposed in [21]. Algorithm partition and parallel processing (using multiple chips) are also indicated . A two-level pipelined VLSI systolic architecture for pattern clustering is proposed in [92]. The architectures for two kinds of pattern matching: string-matching and dynamic time-warp pattern-matching are described in [18]. The algorithm partition problems are discussed and the formal verifications of the proposed VLSI architectures are also given. [21] proposes a VLSI architecture for dynamic time-warp recognition of hand-written symbols. Algorithm partition problems and verification of the proposed architecture have been solved.

The advantages of three-dimensional circuits are studied by comparing sample three-dimensional realizations of certain common families of circuits, namely, permutation networks, FFT circuits, and complete binary trees, with the families' optimal two dimensional realizations [104]. Many books about VLSI systems have been published [85, 64, 102, 98, 122, 119]. There are so many books and articles about VLSI systems, such that the above list is only a part of them.

2.2 VLSI Design Methodology

How to design VLSI architecture systematically has attracted much attention of many researchers. [75] introduces a graph-based mapping methodology for mapping homogeneous and heterogeneous computational graphs onto systolic arrays. [32] proposes a mathematical formalism for the synthesis and qualitative analysis of computational networks that treats data and control in the same manner. [130] presents an overview of an extension to a mathematically based methodology for mapping an algorithmic description into a concurrent implementation on silicon. [130] presents a formalism for describing the behavior of computational networks at the algorithmic level. [74] uses wavefront concept to perform the transformation of algorithms into VLSI structures. [100] present a formal notation and semantics to describe the performance of VLSI systems. [37, 87, 88] present the method of designing algorithms for VLSI systolic array. The mapping procedure is based on the mathematical transformations of index sets and data dependence vectors. [60] describes the method mapping algorithms to VLSI implementation, in particular, the loop reindexing transformation is used. [12] presents a geometric representation of array computation. [70] proposes an algebraic representation, together with a semantics for VLSI algorithm designs. The paper demonstrates its use in the design and verification of systolic algorithm. [100] presents a formal model of linear array processors suitable for VLSI implementation as well as graph representation of programs suitable for execution on such a model. [86] gives a general theory for characterizing and realizing algorithms in hardware. In [79], systolic arrays are characterized by three classes of parameters: the velocities of data flows, the spatial distributions of data and the period of computation. [22, 19]

propose the space-time domain expansion approach to VLSI and the computational model based on this approach. [38] gives a summary on systematic approaches to the design of algorithmically specified systolic arrays.

2.3 PRIP Real-Time Computing

Information scientists have long recognized that: "one picture is worth a thousand words", and over the last decade, extensive research and development has been devoted to pattern analysis and image understanding by computers, practical applications of such computers include the processing of biomedical images for diagnosis; the recognition of characters, fingerprints, and moving objects; remote sensing, industrial inspection; robotic vision; military intelligence; and communications data compression [44].

Many mathematical method have been offered for solving pattern recognition problems, but all are primarily either decision theoretic (statistical) [28] or syntactic (structural) [40]. A block diagram of a pattern recognition system based on this general point of view is shown in Figure 2[45]. Table 3 summarizes the major pattern recognition approaches [45]. A block diagram of a decision-theoretic pattern recognition system and a block diagram of a syntactic pattern recognition system are shown in Figure 3 and Figure 4 respectively [45]. A typical image analysis system consists of four processing stages as depicted in Figure 5 [52].

One difficulty in make PRIP systems practically feasible, and hence more popularly used, is the requirement of computing time and storage. The situation is particularly serious when the patterns to be analyzed are quite complicated [42].

The existing system architectures for PRIP can be classified into three categories: SIMD array processor, pipelined vector processor and MIMD multiprocessor systems [41]. There are several books about computer architectures for PRIP [30, 97, 43].

The advent of VLSI technology has triggered the thought of implementing many image processing and pattern recognition algorithms directly in hardware chips. VLSI structures offer high speed and high reliability - both of which are essential to real-time PRIP tasks. Many VLSI algorithms and their implementation have been developed

for PRIP [11, 116, 10, 63]. These are very important to the real-time computing of PRIP.

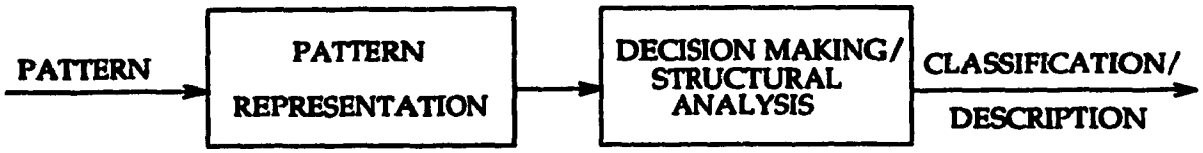


Figure 2: Pattern recognition system.

Table 3: Pattern recognition approaches

APPROACH	PATTERN REPRESENTATION	DECISION MAKING IN TERMS OF SIMILARITY CRITERIA
Template matching	Raw data	Direct matching
Decision theoretic	Feature vector	Discriminant function,
		minimum distance, nearest neighbor, maximum likelihood, minimum Bayes risk, etc.
Syntactic	String Tree Graph	Parsing error-correcting parsing, string matching, tree matching graph matching error-correcting graph matching. etc.

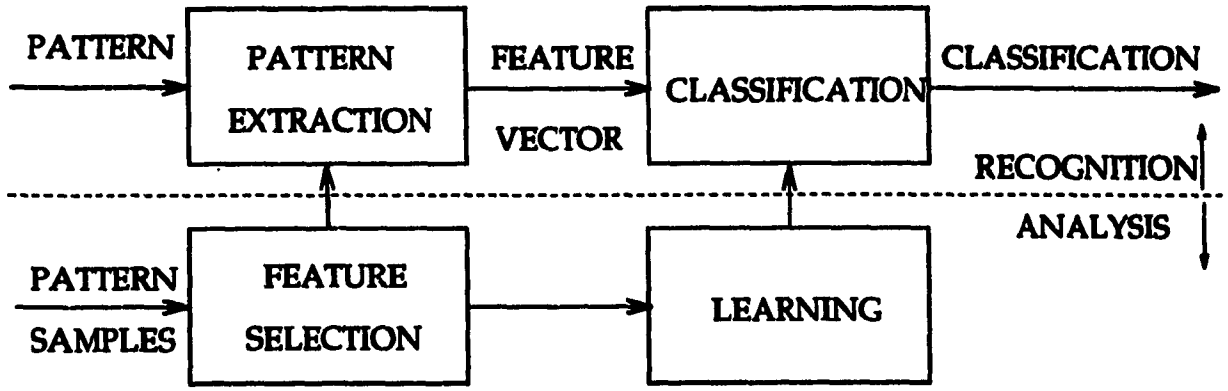


Figure 3: Decision-theoretic pattern recognition systems.

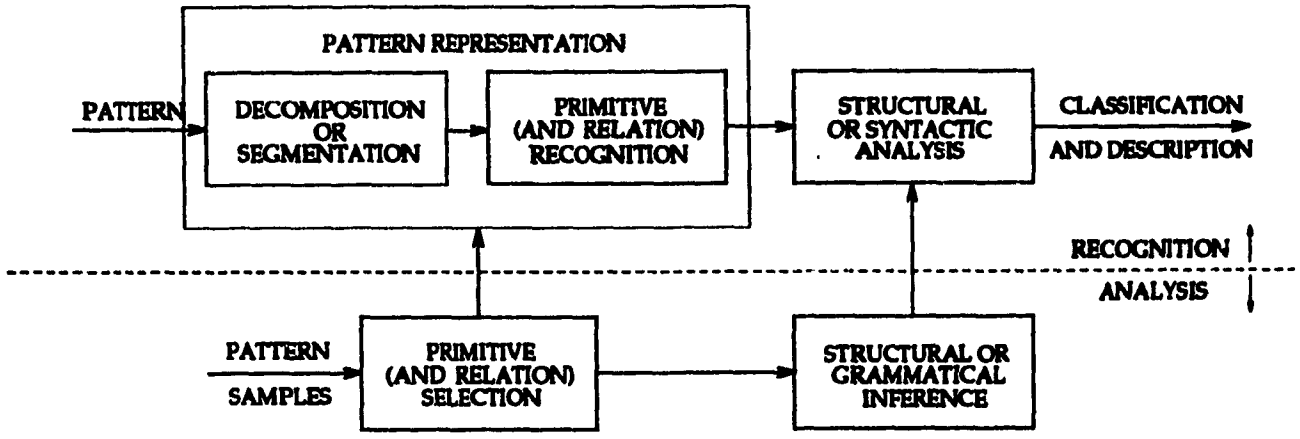


Figure 4: Syntactic pattern recognition system.

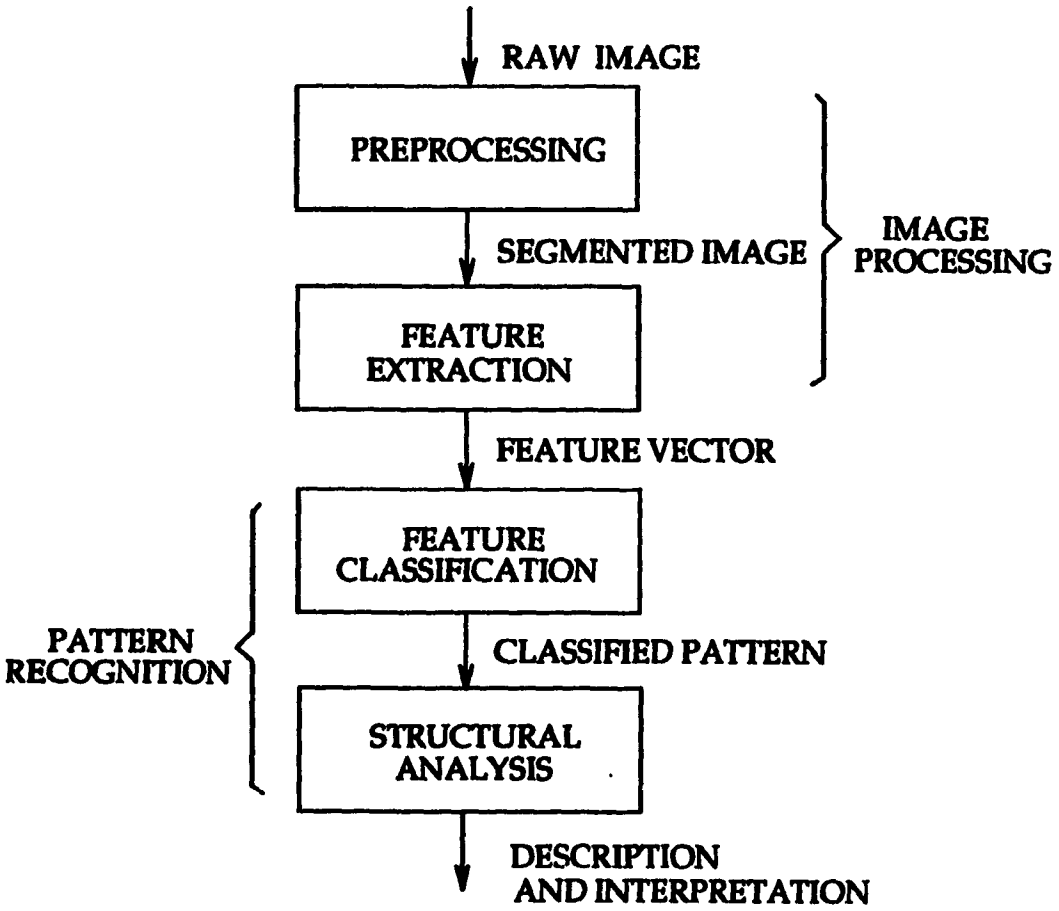


Figure 5: Image analysis system.

2.4 Feature Extraction Methods

Feature extraction is applied in order to concentrate the information in the image into a few, highly selective features. The selection of a stable and representative set of features is the heart of pattern recognition and image processing processing system design.

Features can be generally divided according to the manner in which they are derived. Features which can be derived by local transformations, with little or no regard for the inherent structure of the pattern are called local features. Similarly, method which derived features by global transformations, again with little or no regard for structure are called global features. All other methods pay some attention to the structure of the pattern, and their features are thus classified as structure features.

2.4.1 Local Feature Methods

Local feature methods are distinguished by their relative ignorance of the structure of patterns, and the limited extent of the region used to compute any one feature. Within the category of local features, we find the pixels themselves, the so-called "cellular features", and polygonal approximation of the component boundaries. As the name implies, the pixel feature simply uses the raw pixel data as the feature vector. Cellular features replace each pixel or group of pixels in the image with the result of a local transformation. Polygonal approximation involves fitting a chain of line segments to the outside of the connected regions representing the input, and using list of line segments as the feature vector.

In general, a cellular feature consists of a regular subsampling of the input image, with some local feature being computed at the center of each sample region.

The simplest local feature method uses the pixels themselves as the feature vector. The system merely registers the input image with a template representing the average or "ideal" character in each category, and performs a comparison, or inner product, pixel by pixel. An in-depth study of template matching was carried out by [108].

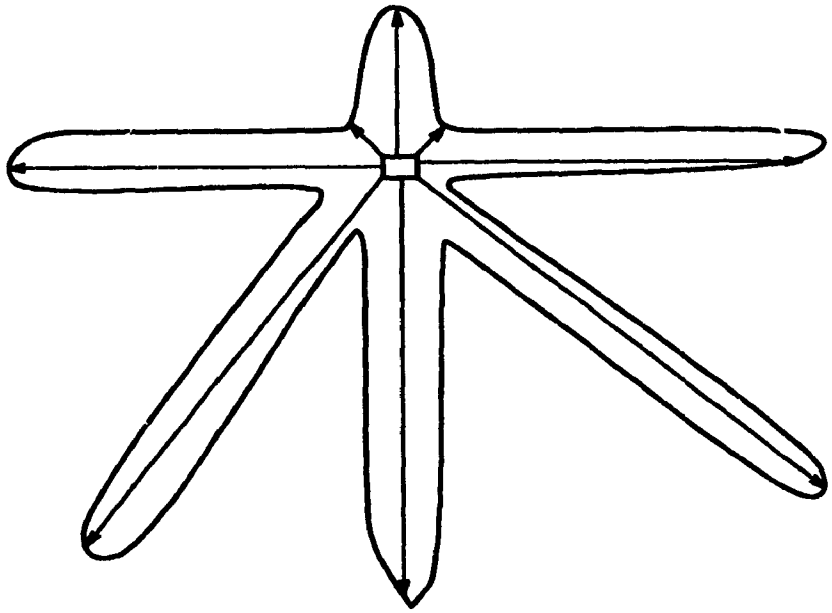
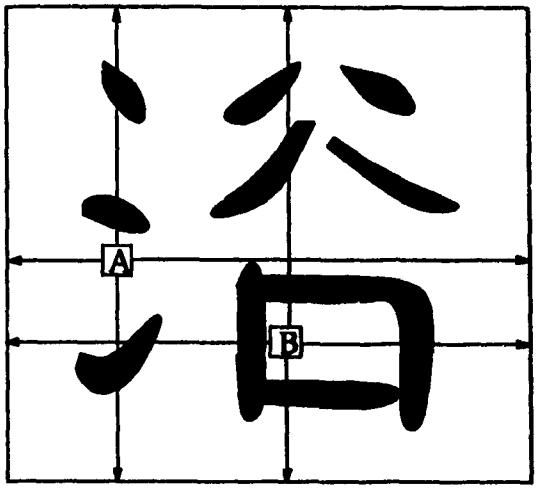


Figure 6: The directional segment strength feature.

Other local features which have been used include *line element gradient, directional segment strength, stroke count*, and *stroke domain*. The line element gradient feature [140] can be considered to be a special case of the directional segment strength feature. The stroke domain feature [135] is a special case of the stroke count feature.

The basic idea of *directional segment strength* [93] is to encode at each pixel position the distance one may travel in each of the eight main directions from that (black) pixel without encountering a white pixel. The *line element gradient* simply records whether or not it is possible to move to the neighboring element in each of eight directions. The directional segment strength feature is shown schematically in Figure 6, where the feature is the length of each of the eight rays extending from the pixel being examined represented by the square.

The stroke count feature counts the number of strokes crossed by a ray from the current point to the edge of the character frame in each of four or eight directions (Figure 7). In [120], strokes are only counted if they are approximately perpendicular to the ray. The stroke domain feature records the existence of strokes both above and below the background element being considers, or to the left and right. As such, it is



(a). locations of pixels A and B.

(b). stroke count features for those two pixels.

Figure 7: The stroke count feature.

a summary of the information available from the stroke count, and may be derived as follows:

$$d_h = \left\{ \begin{array}{ll} 1, & \text{if } c_L c_R > 0 \\ 0, & \text{otherwise} \end{array} \right\}$$

$$d_v = \left\{ \begin{array}{ll} 1, & \text{if } c_A c_B > 0 \\ 0, & \text{otherwise} \end{array} \right\}$$

where d_h and d_v are the horizontal and vertical domain features, respectively, and c_A, c_B, c_L and c_R are the stroke counts above, below, to the left and to the right.

[93, 120] describes a cellular feature which is calculated on a 7×7 grid. The feature consists of eight values at the sample point, being the strength of line segments which are perpendicular to a ray extending from the sample point in each of the eight main directions (Figure 8), where black squares in the feature grid indicate the direction in which there is a perpendicular edge. This feature is a good indicator of the stroke

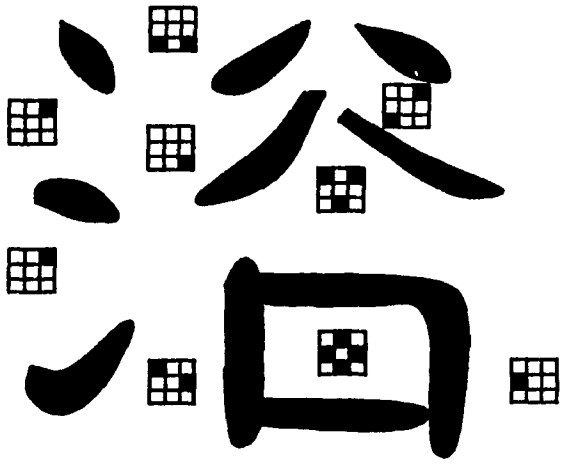


Figure 8: The background cellular feature.

structure of the character. In [140], four pixel maps are formed. A line segment direction (horizontal vertical, left diagonal, right diagonal) is associated with each map, and a pixel appears in a map only if it is a member of a line segment with the correct orientation. The map of each directional feature (Figure 9 and Figure 10) is then repeatedly blurred to remove the effects of slight positional variations. The input pattern is then compared with a standard template using correlation matching. This work is expanded by suggesting the addition of reciprocal feature [139]. This enhancement replaces background pixel elements by values representing the directions in which the element is enclosed. In [137], directional pixel maps like those of [140] are used. Projection profiles are applied to recursively segment the pattern horizontally and vertically. Within each cell in this adaptive grid, a normalized pixel density is recorded for each stroke segment direction. This feature is then blurred and matched with a similarly derived template.

[55] describes a method which uses varying amounts of stroke direction information in a number of different projections. Stroke count, stroke direction, stroke domain and corner feature are extracted. Distributions are formed by horizontal and vertical projections of two halves of the image along 12 scan lines. The projections of the various features are concatenated to form the feature vector. Comparisons are made using correlation matching.

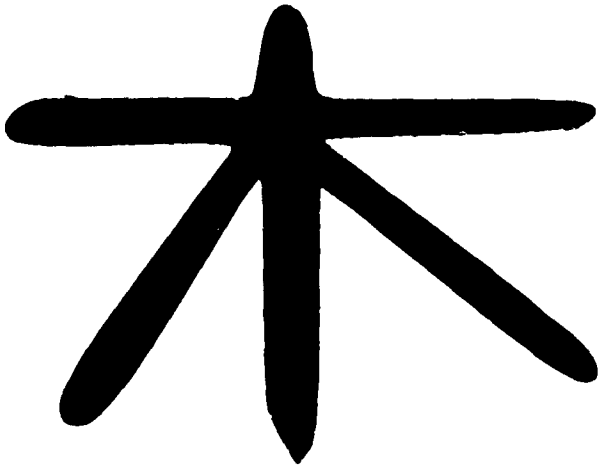


Figure 9: An example Chinese character, "tree".

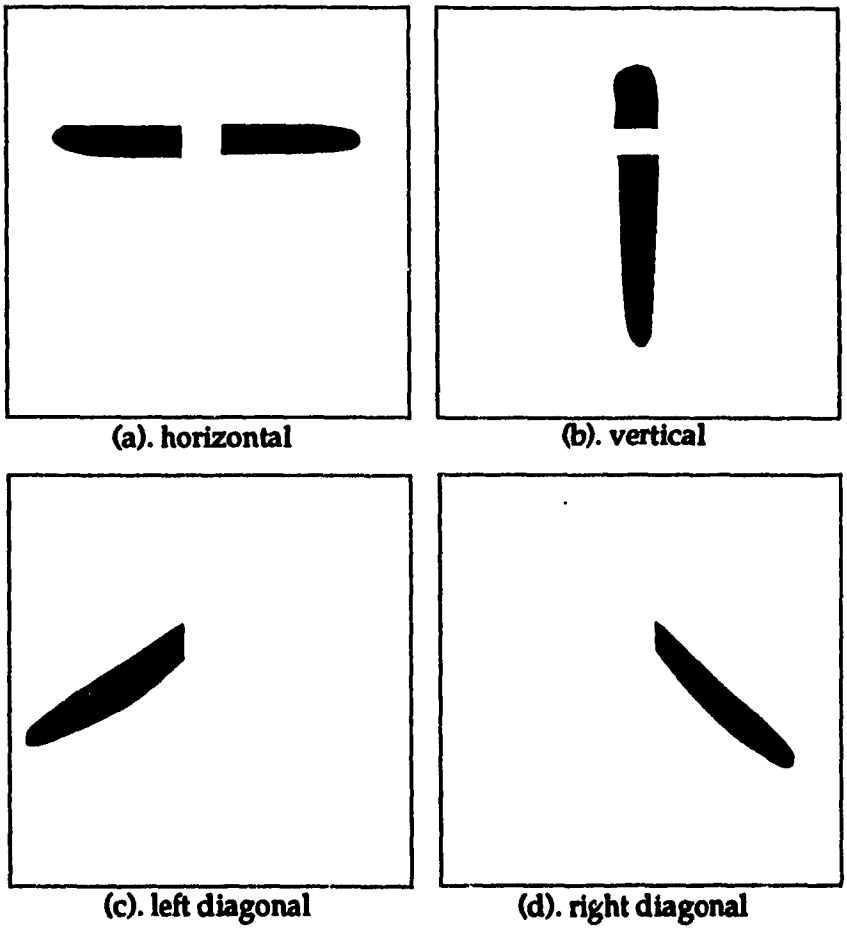


Figure 10: The decomposition of the character into four directional feature maps.

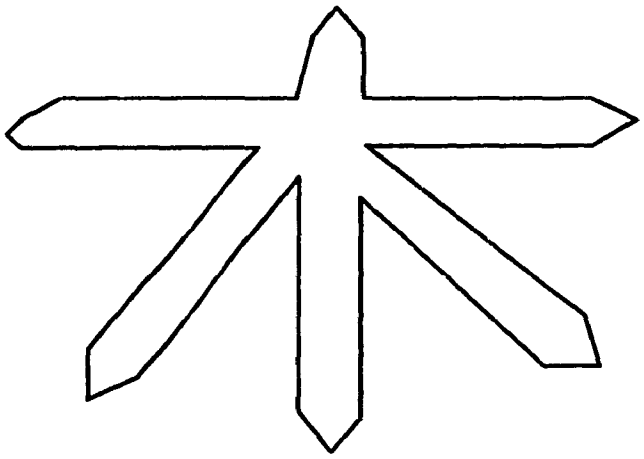


Figure 11: Polygonal approximation of the character in Figure 4..

[144] gives a brief review of cellular features, and introduces the Quartic-Corner-Code (QCC) feature. This feature encodes the shape of each corner of the pattern in terms of an octal number which represents the external profile of the stroke(s) in that corner. The authors then propose to use a combination of directional stroke density (DSD), identification of "over-long" strokes, a peripheral area measure and a stroke density feature to perform fine classification.

Rather than attempting to thin and extract strokes, polygonal approximation represents the perimeter of the black areas of the image with a closed chain of straight line segments (Figure 11) [136]. The advantages of this method is that preprocessing is therefore simplified. Disadvantages include that individual segments in the polygon convey less information than segments of a stroke. This means that the system must be quite tolerant of changes in the length, location, and even existence of a polygonal segment. Yet it must still be able to discern patterns whose categories differ only by the length or location of one stroke.

Another polygonal method, introduced earlier by [134] allows strokes to be split or joined according to a set of rules, so that the stroke configuration in the template can be matched. Perimeters of strokes are approximated by (clockwise) directed line segments. A dynamic programming scheme is used to modify the image data so that it matches a template as closely as possible.

[133] also uses border polygons to describe the shape of pattern components. In

order to simplify the matching step, the pattern is divided into several horizontal strips before extraction of the polygons is performed. This increases resolution and reduces the possibility of incorrect matchings. Fuzzy attributes are used to store the pattern templates, and fuzzy matching is used in the comparison step.

2.4.2 Global Feature Methods

This category contains methods which employ a global transformation to summarize the information in the input image. An ordered subset of the transform coefficients is taken to be the feature vector. Projections are a subclass of global transformations. They are 1D transformations of 2D data. In that these methods ignore the inherent structure of image, they are similar to the local methods, but by considering the image as a whole, they treat the input more abstractly. While being more tolerant of local variations, the transformation has the potential disadvantage that it may suppress the very feature that is needed to distinguish a similar pair.

Global transformations which have been applied to Chinese character recognition include the 2D varieties of the Fourier, Hadamard, and Hough transformations as well as the Rapid transformation [125, 17].

The main characteristic of projection methods is the compression of the data through a projection. Black pixel counts are taken along parallel lines through the image area to generate marginal distributions. The distributions may be of local features as well as simple pixel densities. In [135], three projections are used in combination: pixel density, directional stroke density, and stroke domain. The pixel density profile feature is shown in Figure 12. Directional stroke density counts the pixels in strokes which are aligned with the scan line. Three types of stroke domain are defined by the relationships between neighboring stroke: open, enclosed vertically, enclosed horizontally, and closed.

Several different kinds of projection feature have been used simultaneously in [46], [49]. [90] use the projections of three features, each taken in eight different directions. These are *stroke density function* (SDF), the *direction contributivity density* (DCD), and the *peripheral direction contributivity* (PDC). SDF simply counts the number of strokes crossing each of 12 lines vertically, horizontally, and in the two main diagonal

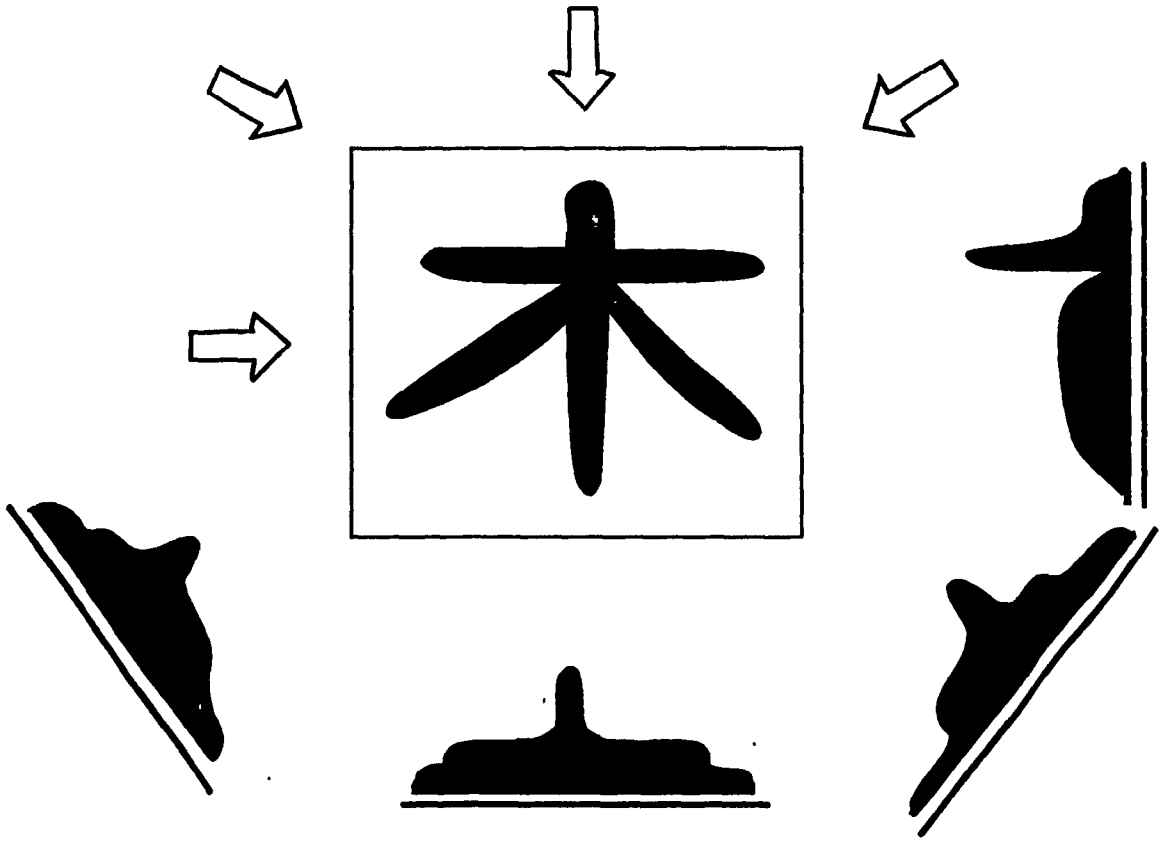


Figure 12: Pixel density (projection) profiles taken in the four primary directions

directions. This feature can be refined by considering each half of the image (split perpendicular to the scan lines). DCD assigns a direction to each stroke, and assigns the stroke direction to each pixel on the stroke. Marginal distributions are taken along 16 scan lines in each of eight directions for each of the four stroke directions. PDC features are formed by assigning stroke directions to pixels and selecting just pixels on the first, second, or third stroke encountered by the scan line. The marginal distribution of such pixels is then taken along 16 lines in eight different directions. In [48] the DCD feature is further refined into Global (G-DCD) and Local (L-DCD) features. In the global form, the scan lines run entirely across the input image. In addition, the angle of the stroke edge in taken into account so that proportional contributions are taken from a stroke which does not align with any of the four axes (horizontal, vertical, and the two diagonals). In the local form, the image area is subdivided into 64 (8×8) subareas, and the DCD is computed on each of these In this case, the directional contribution for each subarea is recorded in four values (horizontal, vertical, left-diagonal, right-diagonal). [2] presents an improvement on the Directional Stroke Density method, by adding two more "global features", which count the number of right angle features in each of four orientations and the number of each of three types of domain (converging, diverging, parallel). Finally, they introduce the Configuration Feature (CF), which summarizes the average and difference angles of the horizontal and vertical stroke pairs surrounding each background domain.

Several different 2D transformations have been applied to the pattern recognition problem. [84] describe a recognition method which uses local transformations to remove nonlinear distortions. Local areas in the input image are Gaussian-sampled and then summarized by transforming them using 2D Hermite polynomials as basis vectors. In [23], a quantized version of the Hough transform, which is called the ChainCode Transform, is studied. Due to the summarizing nature of the transform, it is more tolerant of noise and directional variations than transforms with continuous output maps.

2.4.3 Structure Feature Methods

The third category of features considered is that of structural features. Structural methods assemble a model of the pattern based on the structure which exists in some patterns. The construction is usually bottom-up: from pixel data, line segments or arcs are located: line segments are assembled into strokes; strokes are assembled into pattern components; Pattern components are assembled into a pattern [83]. By examining the way, in which these elements are combined, we get, in essence, a derivation for the pattern. Usually, the derivation is unique; the pattern can thus be identified. The main types of structural feature which have been investigated are *feature points*, or simply *strokes*, *stroke sequence*, and *cure approximation*.

Feature points are defined to be the virtual points of intersection, endpoints, and points of maximum curvature above a certain threshold (Figure 13). At an intersection point, it is sometimes useful to determine which pairs of stroke segments meeting there are continuations of one another in the same stroke. Some methods have been proposed to try to determined this. This general method involves the extraction of so-called "feature points" from the input image. This is a rather primitive method from the standpoint of abstraction: an input is characterized by points at which the curvature of the boundary exceeds a certain threshold. Other feature points may be extracted by local operations. Following extraction, the extrema in the input pattern are matched with the extrema in each standard pattern.

[145] describes a system based on feature points derived from stroke data. Parameters which describe curvature, orientation, length, etc. are included with the feature points that represent the stroke endpoints. A fuzzy-query database is used to implement inexact matching. In [57], the midpoint of a stroke is considered to be another feature point. They add two global features, *Direction Code Distribution* and *Feature Point Distribution*, to stroke data. Input data are thinned, and endpoints, junctions, and inflections are extracted, and stroke segments thus determined. Stroke segments meeting at a junction are connected if the cosine of the angle between them is less than a fixed threshold. Strokes are encoded by direction, length, and midpoint coordinates. [82] describes a recognition method using layers of feature points. Classification is done according to the features in the outer two layers and the number

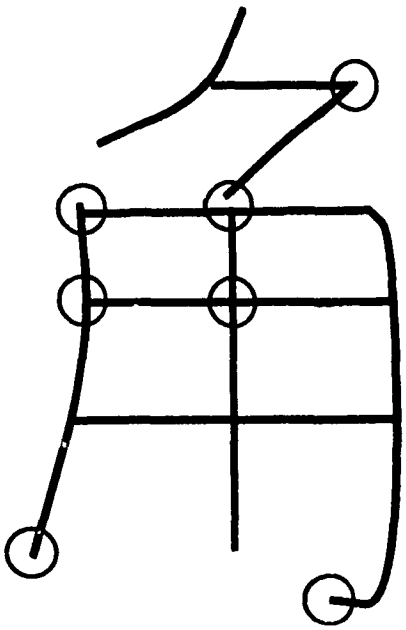


Figure 13: Examples of feature points, indicated by circles.

of intersections lying within those. Another method using fuzzy stroke matching is described by [138]. Character component structure and the maximum horizontal and vertical stroke densities are used for preclassification. Strokes are then extracted with respect to members in the subset of categories identified by preclassification. [14] introduce the idea of a fuzzy-attribute graph (FAG), which combines the ideas of an attributed graph with those of fuzzy set theory. They define an operation which computes the degree of matching between two FAGs, and construct a recognition system based on this idea.

Several researchers have concluded that a sequence of stroke types is a useful feature. [146] describe a method for extracting an ordered sequence of strokes from connected subpatterns. The order is inferred from prior knowledge, and the relationships between connected components can be easily encoded. In [48], preselection is performed using Directional Contributivity Density. Then strokes are extracted from the input image according to each of the 20 or so candidate patterns. A distance measure is calculated, taking into account position and angle differences between input and reference strokes.

Some researchers have found that the information conveyed by a set of line segments is not adequate for recognition. To overcome the difficulty of obtaining a consistent segmentation when presented with the curved strokes in these patterns, they have proposed methods to encode strokes as curves rather than as sequences of line segments. [107, 106] investigated the application of splines to pattern representation. Their method improves on polygonal approximation by replacing straight line segments with smooth spline curves. Rather than force the whole boundary of the pattern to be represented by one spline, critical points are identified (at the intersections and ends of strokes, for instance), and separate splines are run between each neighboring pair. After the splines are calculated, additional critical points may be determined at regions of high curvature, and the spline segments affected are recalculated.

Chapter 3

Regional Projection Transformation

Projection transformations have been proposed and widely used in feature extractions [135, 46, 49, 90, 48, 2]. However, they still encounter great difficulty in dealing with compound patterns containing unconnected patterns and patterns with isolated noises such as those shown in Figure 14.

In the following sections, we propose a new approach called regional projection transformation (RPT). This approach concentrates a compound pattern into an integral object without internal hole. Since the pattern transformed from the RPT contains only one outer contour, it not only makes the contour analysis method become applicable, but also simplifies the process.

We can construct a pattern recognition system using the regional projection transformation (RPT) illustrated in Figure 15. Such system consists of four phases:

(1) Regional projection transformation (RPT),

(2) Contour processing,

(3) Transformation of numerical features.

(4) ISOETRP decision tree.

They can be described briefly below.

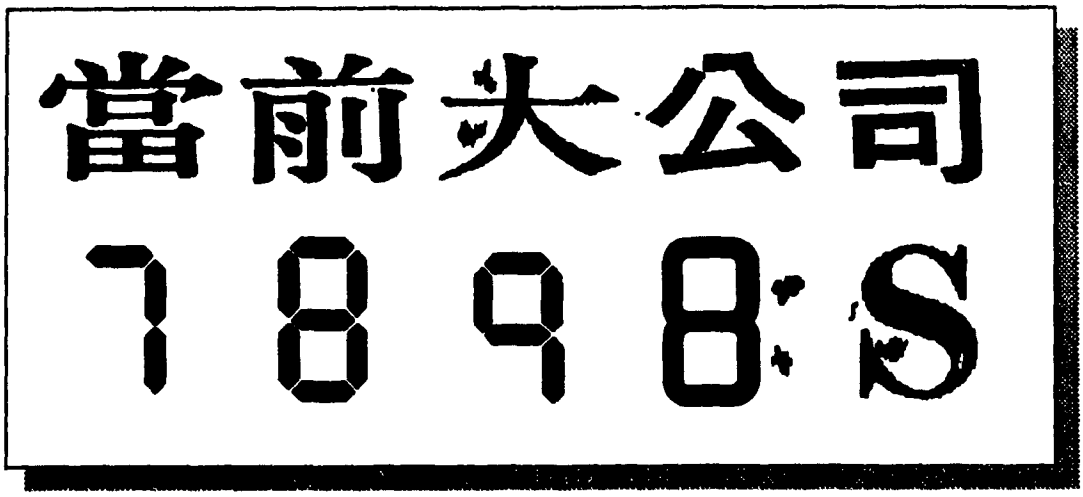


Figure 14: Examples of compound pattern.

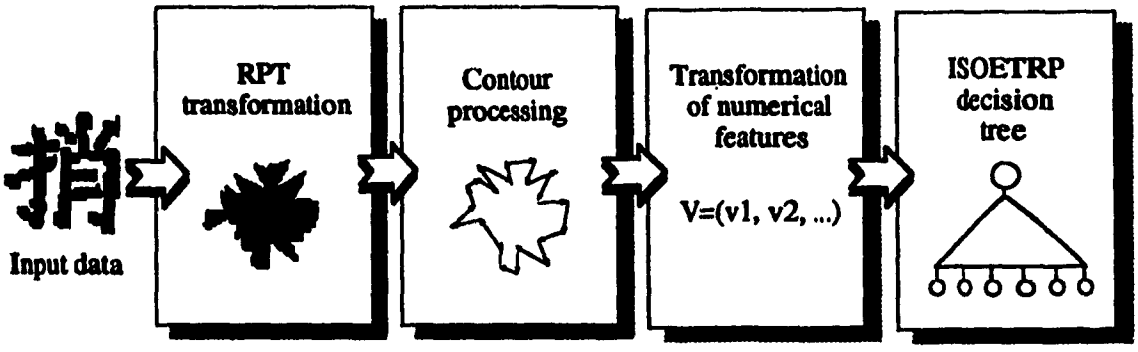


Figure 15: Block diagram of the recognition system using the RPT.

First, a compound pattern has been concentrated into an integral object. Two concentration transformations: (1) diagonal - diagonal regional projection transformation (DDRPT), and (2) horizontal - vertical regional projection transformation (HVRPT) have been proposed in this approach. The patterns transformed by these two schemes are called DDRPT pattern and HVRPT pattern respectively. They possess several important characteristics which facilitate contour analysis. Although the shapes of both DDRPT and HVRPT patterns have been changed, they still contain essential information for their identification. This can be evaluated by n-th moment and experimental results. After the first phase, an integral pattern is generated, and the existing parallel algorithms of contour processing such as those described in [13, 29, 121, 132] can be applied. In this paper, an eight-neighbour technique is used to extract the contour. In the sequel, a contour chain is collected. It can be considered as a cycle function to which a parallel orthogonal transformation, such as FFT [16], can be applied to produce the numerical features to be classified by the ISOETRP decision tree [112]. This thesis will emphasize the first phase of the recognition system, i.e. regional projection transformation which plays a key role in processing compound objects.

A theoretical analysis of the compound and integral patterns will be discussed in the next section. Two RPT's, the DDRPT and HVRPT, will be described and analyzed in Section 2. The patterns transformed by these two transformations possess several important characteristics which can simplify contour processing. These characteristics will also be presented in Section 2.

3.1 Integral Pattern and Compound Pattern

A digitized pattern Ω is composed of a finite set of pixels, $\Omega = \{ \alpha_1, \alpha_2, ..., \alpha_n \}$, which can be viewed as a *weighted graph* [103].

3.1.1 Weighted Graph (WG)

Definition 3.1.1.1 Let $\Omega = \{ \alpha_1, \alpha_2, ..., \alpha_n \}$ be a pattern. The structure of a pattern can be represented by a finite undirected graph termed by *weighted graph (WG)*, which consists of a set of vertices V, a set of edges E, a function f from E to $\{\{u,v\}|u,v \in V, u, v \text{ are adjacent}\}\}$, and a set of weights $D(E)$, such that

$$WG = \{V, E, D(E)\},$$

$$V = \{v_1, v_2, ..., v_n\} = \{\alpha_1, \alpha_2, ..., \alpha_n\},$$

$$E = \{e_1, e_2, ..., e_m\}, \quad D(E) = \{D(e_1), D(e_2), ..., D(e_m)\},$$

$$f(e_i) = \{v_j, v_k\}, \quad D(e_i) = \overline{v_j, v_k}, \quad \forall_i(v_i \in \Omega). \tag{1}$$

Equation (1) implies that each pixel α_i of the pattern Ω is a node v_i of the WG, and edges e_i's are connected among all pairs of adjacent nodes. Each edge e_i=$\{v_j, v_k\}$ is weighted with a distance between its endpoints $D(e_i)$=$\overline{v_j, v_k}$.

3.1.2 Pattern Graph (PG)

Definition 3.1.2.1 A WG is called a *pattern graph (PG)* if all edges with a weight higher than δ have been removed from the WG such that

$$PG = \{V, E, D(E)\}, \quad D(E) = \{D(e_i)| \quad D(e_i) \leq \delta\} \tag{2}$$

where, δ is a threshold depending on the structure of the specific pattern.

3.1.3 Integral Pattern and Compound Pattern

Definition 3.1.3.1 A pattern is called *integral pattern* if its PG is connected. Otherwise it is termed as *compound pattern.*

In fact, compound patterns can be classified into two types: (1) unconnected pattern which is composed of several parts, (2) noise-patch pattern where several isolated patches of noise are embedded in it. By the above definition, Patterns in Figure 14 are compound patterns. Recognizing such compound patterns is much more difficult than the integral ones. To avert this problem, a compound pattern will be transformed into an integral object by the proposed method.

3.2 Regional Projection Transformation (RPT)

The basic principle of the RPT is that all pixels of a pattern are projected onto some base lines termed *projection bases* such as horizontal, vertical, diagonal, etc. It makes sense to concentrate the compound pattern into an integral one. Projection technique has been used to carry out this function.

Let $f(x,y)$ be a pattern, and R stands for an area of the pattern. Assume that $f(x,y) = 0$ lies outside the pattern. $\delta[...]$ denotes a delta function, which is everywhere zero except where its argument is zero, and whose integral from -∞ to ∞ equals one. $t = xsin\varphi - ycos\varphi$ gives the Euclidean distance of a line from the origin. If the projection angle from the x-axis is φ, the projection can be defined as follows [95]:

$$p(\varphi,t) = \sum_{R} f(x,y)\delta[xsin\varphi - ycos\varphi - t]. \tag{3}$$

Once the entire area R has been broken into several smaller regions R_i, $i = 1,2,...,k$, the projection can be divided into several sub-projections which will take place in these regions such that

$$\begin{aligned} p(\varphi,t) &= \sum_{i=1}^{k} p_i(\varphi_i,t_i) \\ &= \sum_{i=1}^{k}\sum_{R_i} f_i(x,y)\delta_i[xsin\varphi_i - ycos\varphi_i - t_i]. \end{aligned} \tag{4}$$

3.2.1 DDRPT

The area of a pattern is divided into 8 sub-regions, R_i $i = 1,2,...,8$. The projection bases are two diagonals (i.e. 45° and 135°), all pixels in the pattern are concentrated onto these two diagonals as shown in Figure 16. The eight sub-regions are symbolized by A, B, C, D, E, F, G, and H such that

$$\begin{aligned} T_{DD} &= \bigcup_{i=1}^{8} T_{DD}^{i} \\ &= A(180°,y) \cup B(270°,x) \cup C(270°,x) \cup D(0°,y) \cup \\ &\quad E(0°,y) \cup F(90°,x) \cup G(90°,x) \cup H(180°,y), \end{aligned} \tag{5}$$

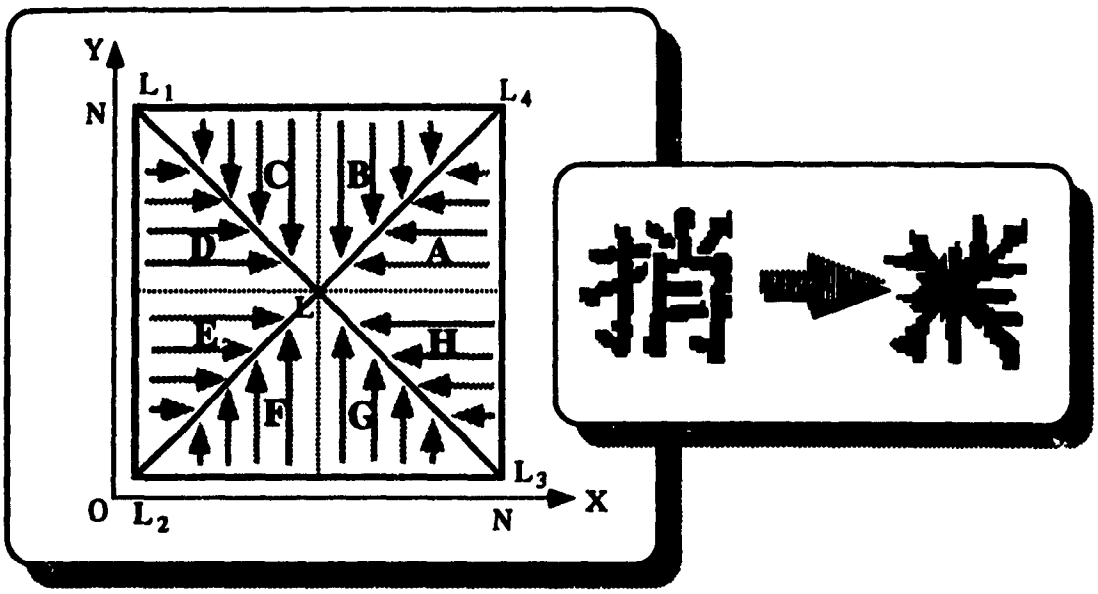


Figure 16: DDRPT.

where, T_{DD} and T^i_{DD} stand for the transformation of the whole pattern and the sub-transformation i respectively. In the concrete, $B(270°, x)$ - $C(270°, x)$ denote sub-transformations for sub-regions starting from B counter clockwise until C respectively.

From equation (3), 8 sub-projections can be calculated as follows:

$$
\begin{aligned}
B(270°, x) &= \sum_{y=x}^{N} f(x,y) && \text{for } x = \frac{N}{2}, \frac{N}{2}+1, ..., N \\
C(270°, x) &= \sum_{y=N-x}^{N} f(x,y) && \text{for } x = 0, 1, 2, ..., \frac{N}{2} \\
F(90°, x) &= \sum_{y=0}^{x} f(x,y) && \text{for } x = 0, 1, 2, ..., \frac{N}{2} \\
G(90°, x) &= \sum_{y=0}^{N-x} f(x,y) && \text{for } x = \frac{N}{2}, \frac{N}{2}+1, ..., N \\
A(180°, y) &= \sum_{x=y}^{N} f(x,y) && \text{for } y = \frac{N}{2}, \frac{N}{2}+1, ..., N. \\
D(0°, y) &= \sum_{x=0}^{N-y} f(x,y) && \text{for } y = \frac{N}{2}, \frac{N}{2}+1, ..., N
\end{aligned}
\tag{6}
$$

$$E(0°, y) = \sum_{x=0}^{y} f(x, y) \qquad \text{for } y = 0, 1, 2, ..., \frac{N}{2}$$

$$H(180°, y) = \sum_{x=N-y}^{N} f(x, y) \qquad \text{for } y = 0, 1, 2, ..., \frac{N}{2}$$

Definition 3.2.1.1 Let $\Im_{DD}$ be a pattern transformed from DDRPT defined by equations (5) and (6), and $\Re$ be a set of projection bases $\Re = \{\beta_1, \beta_2, ..., \beta_n\}$. A DDRPT pattern P_{DD} can be represented by

$$P_{DD} = \Im_{DD} \cup \Re. \tag{7}$$

Here, $n = 4$, i.e. the projection bases are four segments of the diagonals (45° and 135°).

Several DDRPT patterns can be exemplified by Figure 18(b).

3.2.2 HVRPT

Similarly, the area of a pattern is also divided into 8 sub-regions, symbolized by P, Q, R, S, T, U, V, and W. But the bases onto which the pixels to be projected differ from that of the DDRPT. In this scheme, all pixels in the pattern are concentrated onto horizontal and vertical lines, i.e. $y = N/2$ and $x = N/2$, as shown in Figure 17.

$$\begin{aligned} T_{HV} = {} & P(225°, t) \cup Q(225°, t) \cup R(315°, t) \cup S(315°, t) \cup \\ & T(45°, t) \cup U(45°, t) \cup V(135°, t) \cup W(135°, t), \end{aligned} \tag{8}$$

where, T_{HV} stands for the transformation of the whole pattern, $R(315°, t)$ - $Q(225°, t)$ denote sub-transformations for sub-regions starting from R counter clockwise until Q respectively. 8 sub-projections can be calculated below:

$$R(315°, t) = \sum_{x=0}^{N/2} f(x, \sqrt{2}t - x + \frac{N}{2}) \qquad \text{for } t = 0, 1, 2, ..., \frac{N}{2}$$

$$S(315°, t) = \sum_{x=0}^{N/2} f(x, \frac{N}{2} - \sqrt{2}t - x) \qquad \text{for } t = 0, 1, 2, ..., \frac{N}{2}$$

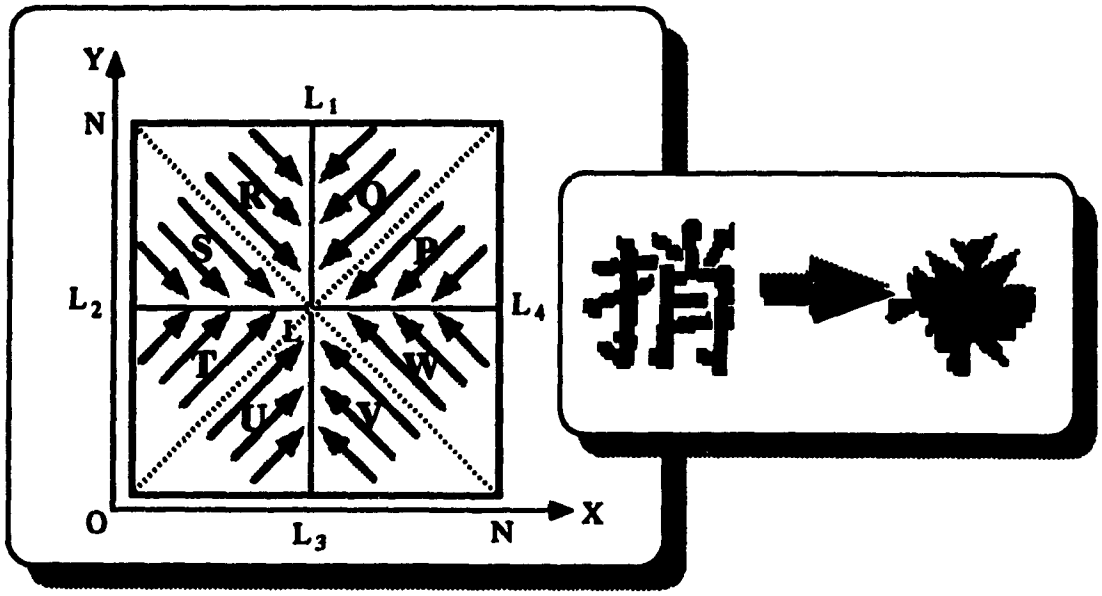


Figure 17: HVRPT.

$$
\begin{aligned}
T(45°,t) &= \sum_{x=0}^{N/2} f(x, x+\sqrt{2}t) && \text{for } t=0,1,2,...,\frac{N}{2} \\
U(45°,t) &= \sum_{x=0}^{N/2} f(x, x-\sqrt{2}t) && \text{for } t=0,1,2,...,\frac{N}{2} \\
V(135°,t) &= \sum_{x=N/2}^{N} f(x, \frac{N}{2}-\sqrt{2}t-x) && \text{for } t=0,1,2,...,\frac{N}{2} \\
W(135°,t) &= \sum_{x=N/2}^{N} f(x, \sqrt{2}t-x+\frac{N}{2}) && \text{for } t=0,1,2,...,\frac{N}{2} \\
P(225°,t) &= \sum_{x=N/2}^{N} f(x, x-\sqrt{2}t+\frac{N}{2}) && \text{for } t=0,1,2,...,\frac{N}{2} \\
Q(225°,t) &= \sum_{x=N/2}^{N} f(x, x+\sqrt{2}t+\frac{N}{2}) && \text{for } t=0,1,2,...,\frac{N}{2}
\end{aligned}
\tag{9}
$$

Definition 3.2.2.1 Let $\Im_{HV}$ be a pattern transformed from HVRPT defined by equations (8) and (9), and $\Re$ be a set of projection bases $\Re = \{\beta_1, \beta_2, ..., \beta_n\}$. An HVRP pattern P_{HV} can be represented by

$$P_{HV} = \Im_{HV} \cup \Re. \tag{10}$$

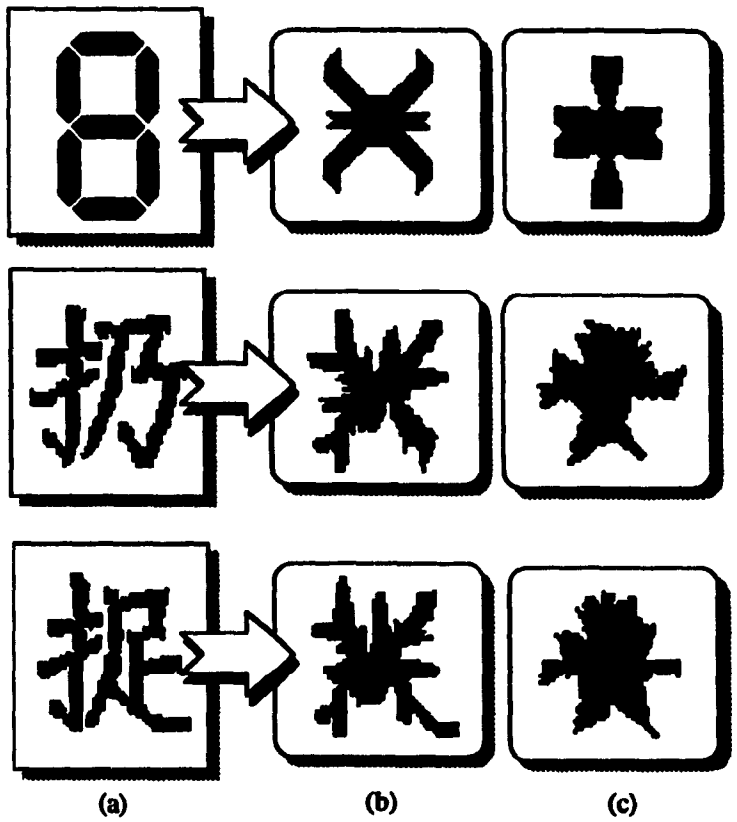


Figure 18: Examples of DDRPT and HVRPT patterns.

Here, $n = 4$, projection bases are two segments of horizontal ($y = N/2$) and two segments of vertical ($x = N/2$).

Several HVRPT patterns can be exemplified by Figure 18(c).

3.2.3 Characteristics of DDRPT and HVRPT Patterns

A couple of very important characteristics of the *DDRPT* and *HVRPT* patterns can be presented in the following theorems, which play a major role in solving the multiple contour problem.

Theorem 3.2.3.1 Either *DDRPT* pattern or *HVRPT* pattern does not contain any hole in it.

Proof: First, we prove the case of the *DDRPT* pattern. A *DDRPT* pattern P_{DD} consists of 8 sub-patterns P_A - P_H, such that $P_{DD} = P_A \cup P_B \cup P_C \cup P_D \cup P_E \cup P_F \cup P_G \cup P_H$. Furthermore, these eight sub-patterns are generated by eight sub-projections given by equation (6). Each sub-projection is made along either a horizontal or vertical line. For instance, the sub-pattern P_C is made by $C(270°, x) = \sum_{y=N-x}^{N} f(x,y)$, for $x = 0, 1, 2, ..., \frac{N}{2}$. It reveals that all pixels in area C are projected onto a diagonal along the horizontal line which produces no interval on the horizontal direction. Thus no hole exists in sub-pattern P_C. Similarly, sub-patterns P_B, P_F and P_G do not contain any hole in them, since sub-projections in areas B, F and G produce no intervals on the horizontal direction. Meanwhile, no intervals along the vertical direction occur in areas A, D, E and H after projecting all pixels along the vertical. Subsequently, the whole *DDRPT* pattern does not contain any hole because all the sub-patterns do not contain any holes.

The case of the *HVRPT* pattern resembles that of *DDRPT*.

□

Theorem 3.2.3.2 Either *DDRPT* pattern or *HVRPT* pattern is a connected pattern

The proof is straightforward, it is omitted here.

Although contour analysis has been amenable to shape recognition, it still has a great difficulty dealing with the *multiple-contour* problem which occurs in two cases:

(1) Compound patterns including unconnected patterns and patterns with isolated noise;

(2) Patterns with internal contours.

The first case can be solved by Theorem 3.2. According to it the compound pattern is concentrated on an integral object to which the contour approach becomes applicable.

The second case can be handled according to the characteristic stated in Theorem 1. The pattern transformed from the *DDRPT* and *HVRPT* contains only one outer

contour. It makes sense to simplify the process.

Chapter 4

Parallel RPT and Its Systolic Implementations

Many pattern recognition and image processing algorithms have been regarded computationally expensive. For example, a two-dimensional convolution with a general 4×4 kernel would require 16 multiplications and 15 additions to be performed for generating each output pixel. To perform this convolution on a 1000×1000 image at the video rate would require the computing power of over 100 MIPS. If the kernel is larger or the dimensionality higher, even more computation power would be required. Consequently their utility in real-time applications is often restricted. With advances in VLSI technology, design and implementation of VLSI systems for pattern recognition and image processing have received a lot of attention [42]. Since numerous problems in pattern recognition and image processing are highly regular computations and offer parallelism in a natural fashion, VLSI architecture and algorithms are need to employ these advantages and efficiently solve these problems in real-time applications. In this chapter, the parallel DDRPT and HVRPT approaches and their implementations on a two dimensional mesh architecture will be described. A parallel contour processing algorithm and its systolic implementation will also be discussed.

4.1 Parallelism in RPT

4.1.1 Parallelism in DDRPT

According to the discussion in Chapter 3, the projection bases used in the DDRPT are two diagonals, and all pixels in a pattern are to be concentrated onto the two diagonals along different projecting directions. The whole pattern has been divided into 8 sub-areas and projection of it can be expressed by 8 sub-projections as shown in equation (5) and (6). Examining equation (5) and (6) finds that the computations of these 8 sub-projections are independent of each other, each of which only needs the values $f_i(x,y)$ of pixels (x,y) within their own sub-areas such that

$$f_i(x,y) \in R_i, \quad R_i \subset R,$$

$$R = \{A(\varphi,t), B(\varphi,t), C(\varphi,t), D(\varphi,t), E(\varphi,t), F(\varphi,t), G(\varphi,t), H(\varphi,t)\}.$$

Therefore, these computations can be done simultaneously. Moreover, because the computation of each sub-projection is the summation of values $f_i(x,y)$ of pixels (x,y) in the same row or column (depending on projection direction of the sub-area). The summations of $f_i(x,y)$ of different rows or columns within the same sub-area can also be done in a parallel fashion. These parallelisms can be described as follows.

algorithm 4.1.1.1

```
Cobegin
        Procedure i (i = A, B, C, D, E, F, G, H);
Coend;
Procedure i (i=A,B,C,D,E,F,G,H)
  Begin
    Parfor j = M1 to M2 do
      Begin
        for k = M3 to M4 do
          Along the vertical direction of Z(Z=X or Z=Y)
          axis, pixel (x,y) in the sub-area i is
```

concentrated onto the segment LLi
with inclination of α

End

End

where,

$$j = \begin{cases} x & i = B,C,F,G \\ y & i = A,D,E,H \end{cases} \qquad k = \begin{cases} y & i = B,C,F,G \\ x & i = A,D,E,H \end{cases}$$

$$Z = \begin{cases} X & i = B,C,F,G \\ Y & i = A,D,E,H \end{cases} \qquad M_1 = \begin{cases} 0 & i = C,E,F,H \\ \frac{N}{2} & i = A,B,D,G \end{cases}$$

$$M_2 = \begin{cases} \frac{N}{2} & i = C,E,F,H \\ N & i = A,B,D,G \end{cases} \qquad M_3 = \begin{cases} 0 & i = D,E,F,G \\ x & i = B \\ y & i = A \\ N-x & i = C \\ N-y & i = H \end{cases}$$

$$M_4 = \begin{cases} x & i = F \\ y & i = E \\ N-x & i = G \\ N-y & i = D \\ N & i = A,B,C,H \end{cases} \qquad \alpha = \begin{cases} 0^0 & i = D,E \\ 90^0 & i = F,G \\ 180^0 & i = A,H \\ 270^0 & i = B,C \end{cases}$$

4.1.2 Parallelism in HVRPT

The same consideration can also be applied to the HVRPT. Projections take place simultaneously in eight sub-areas P, Q, R, S, T, U, V and W . Two horizontal and

vertical lines (i.e. $x = N/2$ and $y = N/2$) are projection bases. The parallel HVRPT approach can be described by the following algorithm:

Algorithm 4.1.2.1

```
Cobegin
    Procedure i ( i = P, Q, R, S, T, U, V, W );
Coend;
Procedure i (i=P,Q,R,S,T,U,V,W)
  Begin
    Parfor x = M1 to M2 do
      Begin
        for y = M3 to M4 do
          Along the vertical directions of diagonals,
          pixel (x,y) in the sub-area i is
          concentrated onto Z axis
          with inclination of α
      End
  End
```

where,

$$Z = \begin{cases} X & i = P,S,T,W \\ Y & i = Q,R,U,V \end{cases} \qquad M_1 = \begin{cases} 0 & i = R,S,T,U \\ \frac{N}{2} & i = P,Q,V,W \end{cases}$$

$$M_2 = \begin{cases} \frac{N}{2} & i = R,S,T,U \\ N & i = P,Q,V,W \end{cases} \qquad M_3 = \begin{cases} 0 & i = T,U,V,W \\ \frac{N}{2} & i = P,Q,R,S \end{cases}$$

$$M_4 = \begin{cases} \frac{N}{2} & i = T,U,V,W \\ N & i = P,Q,R,S \end{cases} \qquad \alpha = \begin{cases} 315^0 & i = R,S \\ 45^0 & i = T,U \\ 135^0 & i = V,W \\ 225^0 & i = P,Q \end{cases}$$

4.2 Systolic Implementation of Parallel RPT

According to the parallelism of the DDRPT and HVRPT, the implementation on a systolic architecture for them is described in the present section. The proposed systolic architecture has the following features: (1) pipelining and multiprocessing techniques are used to perform the computation simultaneously by each pixel; (2) both computation and communication between processing elements are very simple and regular.

First, some definitions that will be used in the following sections are described below:

Definition 4.2.1 In a two dimensional pattern, for pixel (x, y), its neighbours (x+1,y), (x+1,y+1), (x,y+1), (x-1,y+1), (x-1,y), (x-1,y-1), (x, y-1) and (x+1,y-1) are symbolized by β_0, β_1, β_2, β_3, β_4, β_5, β_6, β_7 in counter-clockwise respectively such that

$$\begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \\ \beta_7 \end{pmatrix} \Longleftrightarrow \begin{pmatrix} (x+1,y) \\ (x+1,y+1) \\ (x,y+1) \\ (x-1,y+1) \\ (x-1,y) \\ (x-1,y-1) \\ (x,y-1) \\ (x+1,y-1) \end{pmatrix}.$$

Pixel β_i is called the ith neighbour of pixel (x,y).

Definition 4.2.2 In the two dimensional binary pattern, a pixel is called inner pixel if the following condition is satisfied:

$$(f(x,y)=1) \cap (\forall_{i=0,2,4,6}(f(\beta_i)=1)).$$

It implicates that its own value and all values of its four neighbours β_0, β_2, β_4, β_6, are "1"s.

A pixel is called an outer pixel if the following condition is satisfied:

$$(f(x,y)=0) \cap (\forall_{i=0,2,4,6}(f(\beta_i)=0)).$$

It means its own value and all values of the above four neighbour pixels are "0"s.

Definition 4.2.3 A pixel is an edge pixel if the following condition holds:

$$(f(x,y)=1) \cap (\exists_{i=0,2,4,6}(f(\beta_i)=1)) \cap (\exists_{i=0,2,4,6}(f(\beta_i)=0)).$$

It means that the value of itself is "1", and values of its four neighbour pixels (x+1,y), (x-1,y),(x,y+1) and (x,y-1) are neither all "0"s nor all "1"s.

Definition 4.2.4 For an edge pixel (x,y), if $f(\beta_0)=0$, it is called forward edge pixel (FEP). If $f(\beta_4)=0$, then pixel (x,y) is called backward edge pixel (BEP).

Definition 4.2.5 For an edge pixel (x,y), if $f(\beta_2)=0$, it is called upward edge pixel (UEP) , and if $f(\beta_6)=0$, pixel (x,y) is called downward edge pixel (DEP).

Throughout this paper, it is also assumed that the size of the two dimensional pattern is N × N, and N is odd. If N is even, to make N odd without losing generality, an additional row and column will be added, and values of "0" will be set to these additional pixels.

4.2.1 Machine Model

The proposed systolic architecture, as shown in Figure 19, is not only amenable to the DDRPT and HVRPT but also to parallel contour processing. It is a two-dimensional mesh-connected processor array, consisting of N × N processing elements, one processing element will process one pixel in a two dimensional pattern. There are control signal lines and data buses connecting adjacent processing elements. Each processing element can send or receive data from its eight neighbours through buses. It is assumed that all operations in the system are controlled by the system clock.

The internal structure of each processing element is shown in Figure 20. It has an arithmetic and logical unit (ALU) to perform computations, a control unit to control operations of ALU and a register group (RG).

The RG consists of 21 shifting registers (SR) which are divided into the following four types:

(1) Value register $PI_{x,y}$: It is used to store value f(x,y) of pixel (x,y).

(2) Neighbour registers R[k] ($0 \leq k \leq 7$): They are employed to receive values from its eight neighbour processing elements PE's which correspond to its eight neighbour

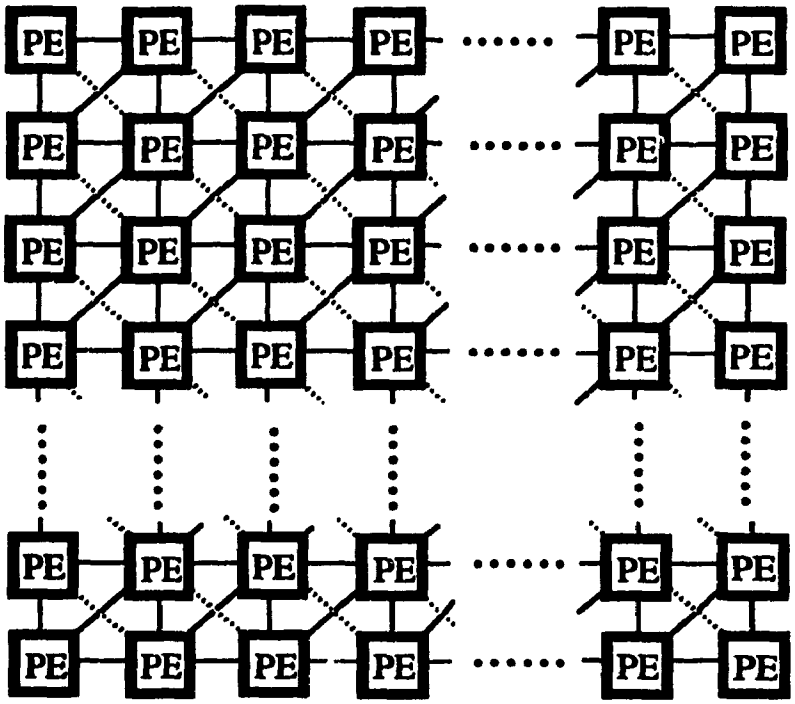


Figure 19: Processor array.

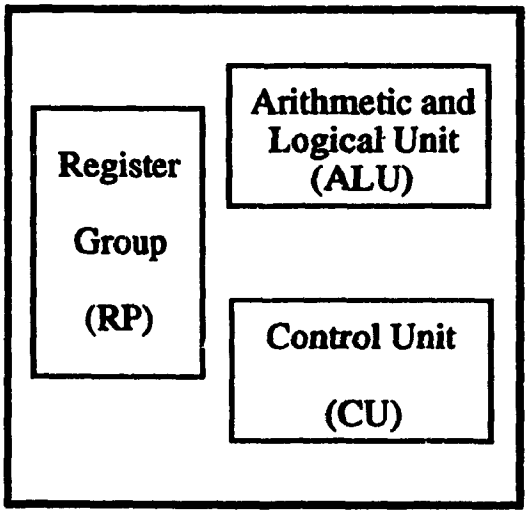


Figure 20: The internal structure of processor element.

pixels β_0, β_1, β_2, β_3, β_4, β_5, β_6, β_7.

(3) Type registers R_{UEP}, R_{DEP}, R_{FEP} and R_{BEP}: They are used to indicate the types of the edge pixels. The contents in registers R_{UEP}, R_{DEP}, R_{FEP} and R_{BEP} denote upward, downward, forward and backward edge pixels respectively. They will be used in parallel contour processing.

(4) Contour-chain registers $R_{en}[k]$, and $R_{ex}[k]$ ($1 \leq k \leq 4$): They record pixels that will be adjacent to pixel (x,y) in the contour chain.

Besides, corresponding to pixel (x,y) in a pattern, each processing element has its own coordinates (x,y) in the processor array. In the following sections, $R_{x,y}$ will be used to denote the register R ($R \in \{R[0] \sim R[7], R_{UEP}, R_{BEP}, R_{DEP}, R_{FEP}, PI\}$ in $PE_{x,y}$.

4.2.2 Systolic Implementation of Parallel DDRPT

It is assumed that all pixels have been loaded into the processor array and the value f(x,y) of pixel (x,y) is stored in the register $PI_{x,y}$ before the computation begins. As discussed in Section 4.1, the projections can be carried out simultaneously in different columns (sub-areas : B, C, F, G) and different rows (sub-areas : A, D, E, H). In our algorithm, such projections are implemented by the concentrations of all pixels towards the projection bases (i.e. two diagonals) from different projecting directions. This procedure consists of the following three steps described below:

(1) After computation has begun, the content of register $PI_{x,y}$ in processing element $PE_{x,y}$ is sent counter projection direction to its adjacent processing element in its area.

(2) For each processing element, the content received from its neighbour processing element is checked. If it is "0", the content of register $PI_{x,y}$ will be sent to the adjacent processing element along the projecting directions in its area, then $PI_{x,y}$ will be reset to "0". Otherwise, a value of "0" will be sent to the neighbour processing element.

(3) After receiving the data sent from its neighbour processing element along the projecting direction, an "OR" operation will be performed in processing element $PE_{x,y}$ using this data and the content of register $PI_{x,y}$ as operand, and the result will be put back to the register $PI_{x,y}$.

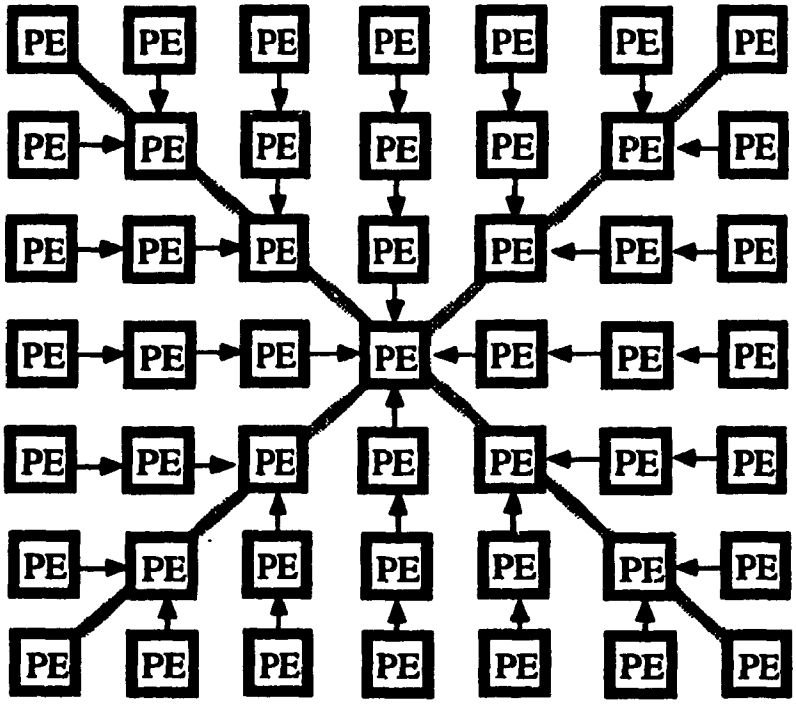


Figure 21: Data flow of parallel DDRPT.

After the above procedure has iterated N times, the concentration processing can be accomplished. All pixels will concentrate towards the projection bases along the projecting directions.

The data flow in parallel DDRPT is shown in Figure 21.

To explain clearly the implementation of the concentration process on a processor array, the concentration in sub-area C is used as an example which is presented as follows:

Step 1: After computation has begun, for every processing element $PE_{x,y}$ in sub-area A, the content of $PI_{x,y}$ is sent to register $R_{x,y+1}[6]$. Then, $PE_{x,y}$ is checked, if $R_{x,y}[6] = 0$, $PE_{x,y}$ sends the content of $PI_{x,y}$ to $R_{x,y-1}[2]$, and $PI_{x,y} = 0$; otherwise, it will send 0 to $R_{x,y-1}[2]$.

Step 2: Operation $R_{x,y}[2] \vee PI_{x,y}$ is executed in $PE_{x,y}$, the result is sent into $PI_{x,y}$.

Step 3: After repeating the above procedure N times, the concentration completes. All pixels in sub-area C are moved towards the concentration base LL1.

Similar procedures can be used to implement the concentrations in other sub-areas. The behaviour of each processing element is described in the following algorithm:

Algorithm 4.2.2.1

This algorithm is executed by each $PE_{x,y}$ in parallel.

```
Begin
  For i = 1 to ⌊N/2⌋ do
    Begin
      send ( PI_{x,y} ,m) ;
      receive (R[q],q) ;
      If R[q] = 0 then
        send (PI_{x,y} , q) ;
      else
        send (0 , q) ;
      receive ( R[m] , m) ;
      PI_{x,y} ← R[m] ∨ PI_{x,y} ;
    End
End
```

where

$$m = \begin{cases} 0 & PE_{x,y} \in \text{areas A, H} \\ 2 & PE_{x,y} \in \text{areas B, C} \\ 4 & PE_{x,y} \in \text{areas D, E} \\ 6 & PE_{x,y} \in \text{areas F, G} \end{cases} \qquad q = \begin{cases} 0 & PE_{x,y} \in \text{areas D, E} \\ 2 & PE_{x,y} \in \text{areas F, G} \\ 4 & PE_{x,y} \in \text{areas A, H} \\ 6 & PE_{x,y} \in \text{areas B, C} \end{cases}$$

Assume that the following operations can be completed within a time unit, in each processing element:

(a) All data passing operations between processing elements,

(b) Arithmetic and logical operations.

The time complexity of algorithm 4.3 can be calculated in the following manner:

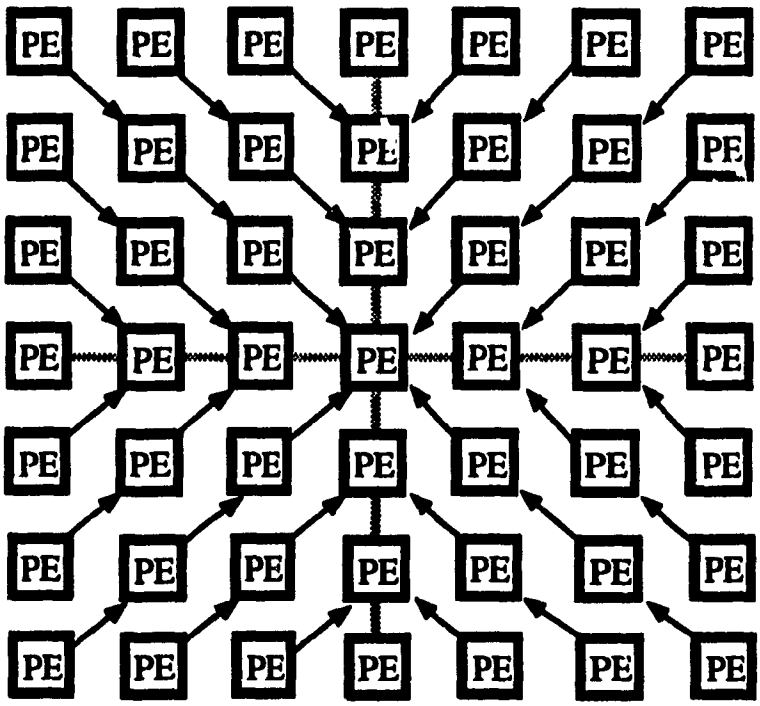


Figure 22: Data flow of parallel HVRPT.

Concentrations of pixels can be done simultaneously not only in different sub-areas of the entire pattern but also in different columns or rows within the same sub-region, since such concentrations within a sub-area are performed along columns (areas B, C, F, G) or rows (areas A, D, E, H). Therefore, the time complexity of algorithm 4.3 is O(N).

4.2.3 Systolic Implementation of Parallel HVRPT

The proposed processor array can also be used to implement parallel HVRPT. As discussed in Section 4.1, projections in HVRPT are also carried out by concentrating all pixels towards the projection bases from different projection directions in eight sub-regions. Its systolic implementation is similar to that of parallel DDRPT algorithm except the directions of data flows and control flows. An example of data flow for it is shown in Figure 22. The projection bases are changed to $x = N/2$ and $y = N/2$ in HVRPT, the directions of the projection are 45°, 135°, 225° and 315°.

The behaviour of each processing element for implementing the parallel HVRPT can be described as follows.

Algorithm 4.2.3.1

This algorithm is executed by each $PE_{x,y}$ in parallel.

```
Begin
    For i = 1 to ⌊N/2⌋ do
        Begin
            send ( PI_{x,y} ,m) ;
            receive (R[q],q) ;
            If R[q] = 0 then
                send (PI_{x,y} , q) ;
            else
                send (0 , q) ;
                receive ( R[m] , m) ;
                PI_{x,y} ← R[m] ∨ PI_{x,y} ;
        End
End
```

where

$$m = \begin{cases} 1 & PE_{x,y} \in \text{areas } P, Q \\ 3 & PE_{x,y} \in \text{areas } R, S \\ 5 & PE_{x,y} \in \text{areas } T, U \\ 7 & PE_{x,y} \in \text{areas } V, W \end{cases} \qquad q = \begin{cases} 1 & PE_{x,y} \in \text{areas } T, U \\ 3 & PE_{x,y} \in \text{areas } V, W \\ 5 & PE_{x,y} \in \text{areas } P, Q \\ 7 & PE_{x,y} \subset \text{areas } R, S \end{cases}$$

Similarly, the time complexity of algorithm 4.4 is also O(N), since pixels can be processed simultaneously not only in different sub-regions of whole object, but also in different concentration lines within each sub-area.

4.3 Parallel Contour Processing

As stated in the previous sections, after concentrating all pixels of a pattern towards the base lines by either the DDRPT or HVRPT, a condensed object that contains only single boundary can be obtained. Although the existing contour processing algorithms such as [13, 29, 121, 132] can work on this type of patterns, a simpler one is competent for such simple contour. Thus, a new parallel contour processing algorithm which is much simpler than others has been developed and its systolic architecture has also been designed. They will be described in this section.

The contour of a pattern can be expressed by a chain consisting of edge pixels described in definitions 4.3 - 4.5. Using a processor array proposed in Section 4.2, all edge pixels based on definition 4.3 can be easily extracted. But such edge pixels will appear in a random order. We still have to find the relationships among them within the pattern and construct a chain that keeps the relationships of adjacent pixels in the pattern. In this section, a parallel algorithm is proposed which carries out both contour extraction and chain establishment. The proposed algorithm has the following feature : each processing element $PE_{x,y}$ in the two dimensional processor array only needs local data (the contents of PI registers in its neighbour processing elements) to identify the adjacent pixels of pixel (x,y) in the contour chain within constant time units.

The main operations which are performed during contour tracing in the proposed algorithm can be presented below:

(1) Distinguishing the type of pixel.

In each processing element $PE_{x,y}$, the content of its register $PI_{x,y}$ (i.e. f(x,y), the value of pixel (x,y)) is sent to its eight neighbour processing elements. Values of its neighbour pixels are also received from its eight neighbour processing elements. They are stored in the registers $R_{x,y}[0] \sim R_{x,y}[7]$. According to the definitions in Section 4.2, the type of pixel (x,y) can be identified by these values and the content of register $PI_{x,y}$ in each processing element $PE_{x,y}$.

(2) Establishing chain consisting of the edge pixels.

If a pixel (x,y) is an edge pixel and belongs to type k (here, k $\in$ { FEP, BEP,

UEP, DEP }), then the following procedure is executed in processing element $PE_{x,y}$ to find the neighbours of pixel (x,y) in the chain.

If pixel (x,y) belongs to the type of UEP, for processing element $PE_{x,y}$, first, the values of its neighbours will be checked counter clockwise starting the 4th neighbour β_4, until the value of one of its neighbours with "1" has been found. Then, the values of other neighbour pixels will be checked clockwise from β_4, until another neighbour pixel with value "1" has been found. These two pixels are neighbours of pixel (x,y) in the contour chain.In the sequel, the coordinates of them will be recorded.

In the case that pixel (x,y) belongs to other types, a similar procedure can be used to search its adjacent pixels in the contour chain except that the starting points differ according to the different types, i. e. starting from β_0 in DEP, β_6 in BEP, β_2 in FEP. Because all values of neighbour pixels have been sent to $PE_{x,y}$ and stored in registers $R_{x,y}[0] \sim R_{x,y}[7]$ in turn in the first operation, the above search procedure can be performed in $PE_{x,y}$ by checking the contents of these registers.

An example of parallel extracting contour and establishing chain is illustrated in Figure 23.

The behaviour of each processing element during contour tracing is described in the following procedure, the code is executed by each processing element $PE_{x,y}$ in parallel.

Algorithm 4.3.1

```
/* STEP (1) : Distinguish type of each pixel */
        Cobegin
          for  m = 0,1,2,3,4,5,6,7 do
            Begin
              send (PI[i,j], m);
              receive (R[m],m1) ;
            End /* for m */
          if  R[0] ∧ R[2] ∧ R[4] ∧ R[6] ∧ PI[i,j] = 0 then
            R_outer = 1;
          if R[0] ∨ R[2] ∨ R[4] ∨ R[6] ∨ PI[i,j] = 1 then
            R_ineer = 1 ;
```

```
            if R_outer ∨ R_ineer = 0 then
               R_edge = 1 ;
            if (R[4]=0) and ( R_edge = 1) then
               R_UEP = 1 ;
            if (R[0]=0) and ( R_edge = 1 ) then
               R_DEP = 1 ;
            if (R[2] = 0)and ( R_edge = 1 ) then
               R_FEP = 1 ;
            if (R[6] = 0) and ( R_edge = 1 ) then
               R_BEP = 1 ;
         Coend
/* STEP (2) : find a chain with clockwise direction */
         Cobegin
            for k = UEP,DEP,FEP and BEP do
               if R_k = 1 then
                  Begin
                     m = p ;
                     while (R[m] = 0) do
                        Begin
                           m = m + 1 ;
                           if m = 8 then
                           m = 0 ;
                        End
                     R_k,enter = m ;
                     m = p ;
                     while (R[m] = 0 ) do
                        Begin
                           m = m - 1 ;
                           if m = -1 then
                              m = 7 ;
                        End;
```

$R_{k,exit} = m$;

End

Coend

where,

$$p = \begin{cases} 0 & k = DEP \\ 2 & k = FEP \\ 4 & k = UEP \\ 6 & k = BEP \end{cases}$$

The time complexity of the proposed parallel contour processing algorithm can be computed as follows :

Time complexity = time units needed for step (1) + time units needed for step (2) = O(17) + 4 × O(10) = O(c).

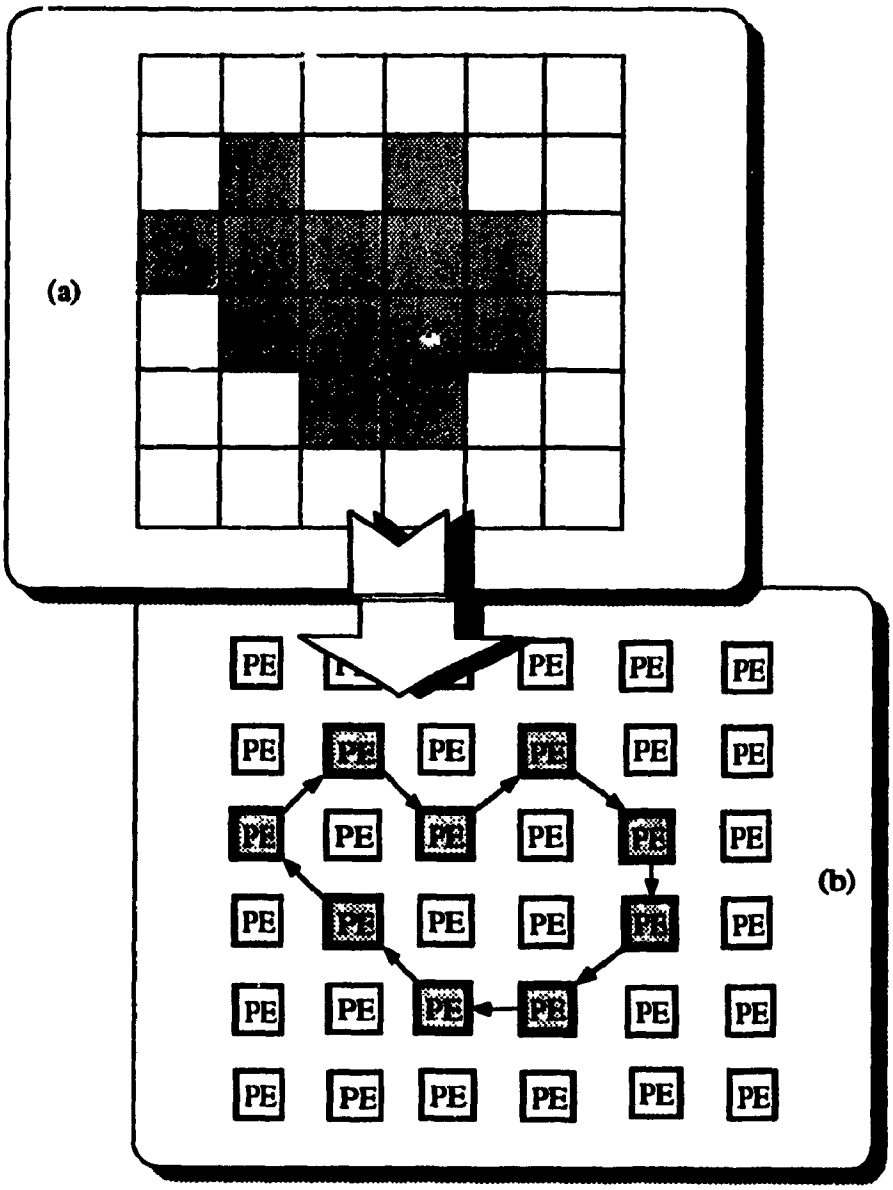


Figure 23: Contour extraction.

Chapter 5

Parallel RPCT and Its Systolic Implementation

In chapter 4, we discussed the parallel regional projection transformation approach and its systolic implementation. This approach consists of two steps. First, it concentrates a compound pattern into an integral object through the projection. Then a parallel contour processing algorithm is used to collected the contour chain of the pattern. However, these two steps are actually independent of each other, these two operations can be done concurrently, yielding a higher degree of parallelism. In this chapter, we introduce a combinatory parallel approach called *regional projection contour transformation* (RPCT), which is the further improvement of parallel RPT. We also discuss its systolic implementation on a linear processor array using a systematic methodology called *canonical mapping methodology* proposed in [75].

5.1 Systolic Array for DDRPCT

To achieve the highest performance of computation, systolic architecture must be carefully designed. In this study, a *canonical algorithm-array mapping* methodology has been used [75]. It consists of three steps:

(1) DG design - mapping algorithm to DG;

(2) SFG design - mapping DG to SFG;

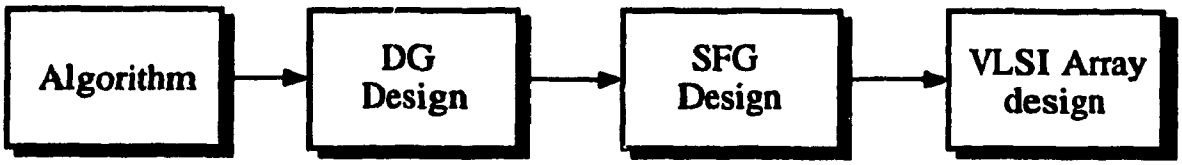


Figure 24: Block diagram of algorithm-array mapping approach.

(3) Array processor design - mapping an SFG into an array processor.

The key concept of this approach can be summarized in a block diagram illustrated in Figure 24.

5.1.1 Algorithm-Array Mapping in Sub-DDRPC 'C'

In order to design an efficient systolic architecture, the data dependencies in the computations must be carefully investigated, and a *dependence graph (DG)* is used. The DG is a graph that shows the dependence of the computations that occur in an algorithm.

A DG for sub-DDRPCT in sub-region 'C' is shown in Figure 25(a), where sub-region 'C' has a size of $\frac{N}{2} \times \frac{N}{2}$. This is a local DG, since all its nodes are dependent on the neighboring nodes located on the lines with 90° only, it contains no global communications in it.

To determine a valid array structure for an algorithm, one straightforward design is to designate one processing element (PE) for each node in a DG. However this design generally leads to very inefficient utilization of the PEs, since each PE can be active only a small fraction of the computation. In order to improve PE utilization, it is often desirable to map the nodes of the DG into a fewer number of PEs. To achieve this, it is useful to map the DG first into an intermediate form termed *signal flow graph (SFG)*.

A SFG is a directed graph consisting of nodes and edges weighted edge delays. The nodes model computations and the edges model communications. A complete SFG

description includes both functional and structural description parts. The functional description defines the behavior within the nodes, while the structural description specifies the interconnection between the nodes. Hence, the SFG can model the implementation, in time and space, of a computation described by a local DG. To obtain this implementation, the DG is mapped into the SFG.

DG-SFG Mapping in Sub-DDRPCT 'C'

The DG-SFG mapping consists of two phases, processor assignment and scheduling, which are described briefly as follows:

Processor Assignment:

1) Assigning the nodes in the index space of the DG to particular nodes of the SFG.

2) A criterion might be to minimize communication of data between processors.

3) A projection method can be applied, in which nodes of the DG along a straight line are assigned to a common PE.

4) The projection maps the DG into a lower dimensional lattice of points, known as the processor space.

5) Mathematically, a linear projection is often represented by a projection vector $\vec{d}$.

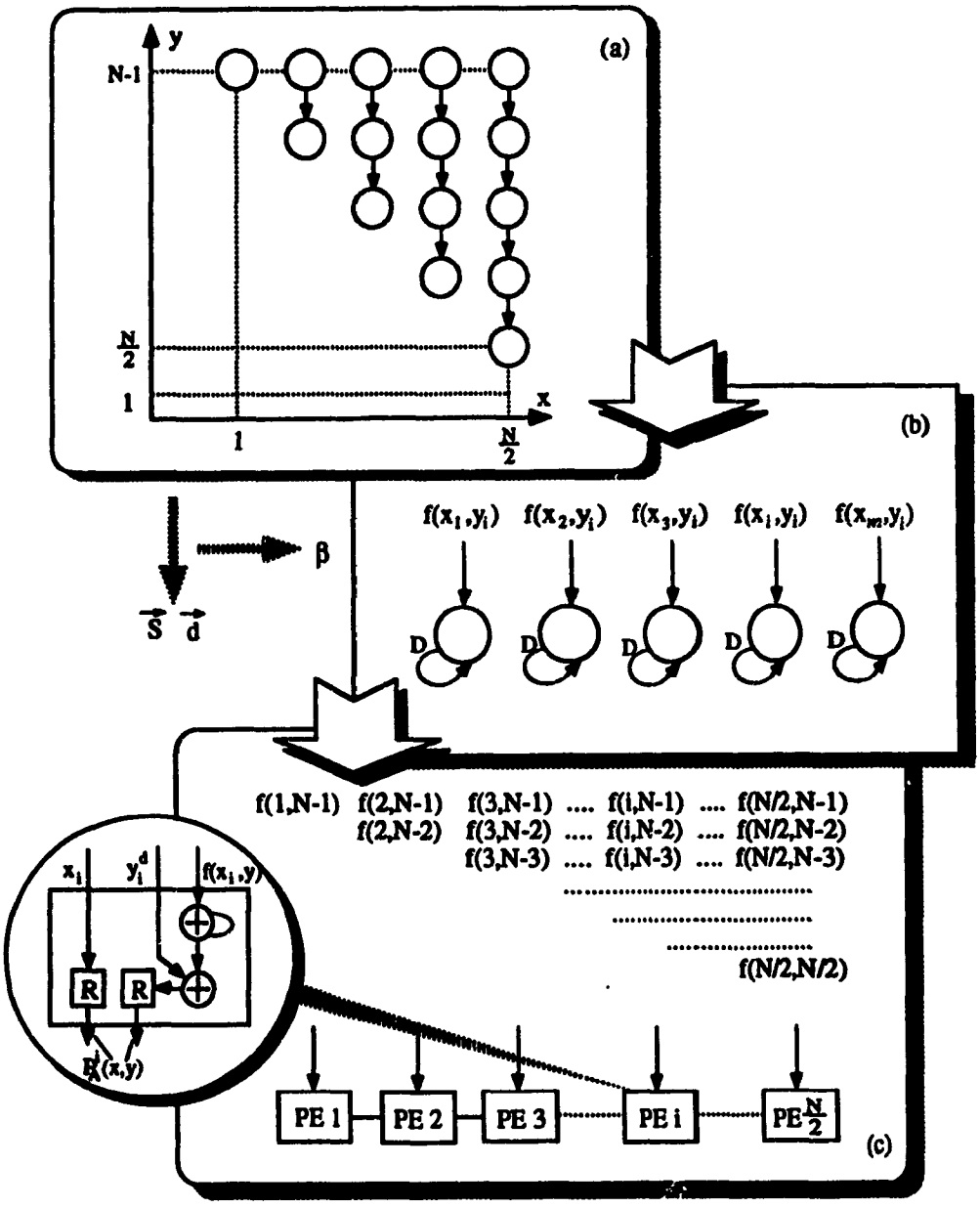
Scheduling:

1) Scheduling the order in which these nodes are to be computed.

2) A criterion might be to minimize total computing time.

3) A schedule function represents a mapping from the N-dimensional index space of the DG into a 1-D schedule (time) space.

4) A linear schedule is based on a set of parallel and uniformly spaced hyperplanes in the DG.



(a) DG, (b) SFG, and (c) VLSI architecture

Figure 25: Algorithm-array mapping in subdiagonal-diagonal RCPT'C'.

5) Mathematically, the schedule can be represented by a column schedule vector $\vec{s}$, pointing to the normal direction of the hyperplanes.

In this study, *algebraic approach* has been used [75]. In general, it involves three steps: (1) node mapping, (2) arc mapping, and (3) input/output mapping.

1 Node mapping

This mapping assigns the node activities in the DG to the processors. For any projection direction, a processor space is orthogonal to the projection direction. A processor array can be obtained by projecting the index points to the processor space. Given a DG and a projection vector $\vec{d}$, the most likely used schedules for the SFG projection are (1) default schedule, (2) recursion schedule, (3) systolic schedule, and (4) optimized schedule [75]. In this thesis, a default schedule is used, which can be defined below.

Definition 5.1.1.1 Let $\vec{\lambda}$ be a hyperplane vector of a DG, $\vec{d}$ a projection vector, a schedule is called a *default schedule* if the following condition is satisfied:

$$\vec{\lambda}^T \vec{d} = 0$$

which implicates that the hyperplanes are orthogonal to the projection direction. In other words, the normal direction of hyperplanes is parallel to the projection direction.

Consider the DG of sub-DDRPCT 'C' as shown in Figure 25(a), the projection vector $\vec{d}$, schedule vector $\vec{s}$ and processor basis β are computed as follows:

$$\vec{d} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}.$$

Since the default schedule has been used, the schedule vector is parallel to the projection direction

$$\vec{s} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}.$$

The processor basis β is orthogonal to $\vec{d}$, i.e.

$$\beta^T \vec{d} = 0$$

and it can be calculated as

$$\beta = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Therefore, the node mapping can be obtained

$$\beta^T \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}^T \begin{bmatrix} x \\ y \end{bmatrix} = [x].$$

2 Arc mapping

This operation maps the arcs of the DG to the edges $\vec{e}$ of the SFG. The arcs in the DG are replaced with edges $\vec{e}$ associated with zero/nonzero delay $D(\vec{e})$ in its corresponding SFG. The number of delays on each edge depends on the number of time steps needed.

In the x direction:

$$\begin{bmatrix} D(\vec{e}_x) \\ \vec{e}_x \end{bmatrix} = \begin{bmatrix} \vec{s}^T \\ \beta^T \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In the y direction:

$$\begin{bmatrix} D(\vec{e}_y) \\ \vec{e}_y \end{bmatrix} = \begin{bmatrix} \vec{s}^T \\ \beta^T \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

3 Input/output mapping

After each node of the DG has been projected to a PE, considering each input/output data is connected to some nodes, it is possible to attach the input/output data to their corresponding processors. The SFG node position $\Im$ and time $T(\gamma)$ for input/output can be computed.

Input:

$$\begin{bmatrix} T_i(\gamma) \\ \Im_i \end{bmatrix} = \begin{bmatrix} \vec{s}^T \\ \beta^T \end{bmatrix} \begin{bmatrix} x \\ -1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ x \end{bmatrix}$$

Output:

$$\begin{bmatrix} T_o(\gamma) \\ \Im_o \end{bmatrix} = \begin{bmatrix} \vec{s}^T \\ \beta^T \end{bmatrix} \begin{bmatrix} x \\ -1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ x \end{bmatrix}$$

Once the above mappings have finished, a SFG for sub-DDRPCT in sub-region 'C' can be obtained as shown in Figure 25(b).

Systolic Architecture for Sub-DDRPC 'C'

From this SFG, a systolic architecture with linear array can be designed as Figure 25(c). The input data flow $f(x,y)$ is shown in this figure, the output $P_C(x,y)$ of the array is

$$P_C(x,y) = (P_C^1, P_C^2, ..., P_C^i, ..., P_C^{\frac{N}{2}})$$

According to the following theorem, it is clear that the coordinates of the partial contour points of the sub-DDRPCT pattern can be represented by such outputs.

Theorem 5.1.1.1 Let $P_C^i(x,y)$, $i = 1,2,..., \frac{N}{2}$ be outputs of the processors of the systolic array shown in the above. The values of coordinates of the points in the partial contour of sub-DDRPCT 'C' can be described by these outputs, such that

$$P_C^i(x,y) \ : \ P(x,y) \begin{cases} x = i \\ y = (N-i) + \sum_{y=N-i}^{N-1} f(x,y) \end{cases} \tag{11}$$

Proof: From the systolic array shown in Figure 25(c), the output of each PE can be listed below:

$$\begin{aligned}
&PE_1 \ : \ P(1,y_1) && y_1 = N-1 \\
&PE_2 \ : \ P(2,y_2) && y_2 = (N-2) + \sum_{y=N-2}^{N-1} f(2,y) \\
&PE_3 \ : \ P(3,y_3) && y_3 = (N-3) + \sum_{y=N-3}^{N-1} f(3,y) \\
&\cdots\cdots\cdots\cdots && \\
&PE_i \ : \ P(i,y_i) && y_i = (N-i) + \sum_{y=N-i}^{N-1} f(i,y) \\
&\cdots\cdots\cdots\cdots && \\
&PE_{\frac{N}{2}} \ : \ P(\frac{N}{2}, y_{\frac{N}{2}}) && y_{\frac{N}{2}} = (\frac{N}{2}) + \sum_{y=\frac{N}{2}}^{\frac{N}{2}} f(\frac{N}{2},y)
\end{aligned} \tag{12}$$

From the above list, a claim about the general formula in sub-region 'C' can be written as

$$PE_i \ : \ P(x,y) \begin{cases} x_i = i \\ y_i = (N-i) + \sum_{y=N-i}^{N-1} f(i,y) \end{cases} \tag{13}$$

Now, the *induction proof* is used to verify the above claim.

(a) **Basis** : The case k=1. Immediately the output of processing element PE_1 can be calculated by

$$PE_1 \ : \ x_1 = 1, \qquad y_1 = N - 1$$

(b) **Induction Hypothesis** : Assume in case of k, the output of processing element PE_k is presented by

$$\begin{aligned} PE_{k+1} : \quad x_k &= k \\ y_k &= \underbrace{(N-k)}_{y_k^1} + \underbrace{\sum_{y=N-k}^{N-1} f(x,y)}_{y_k^2} \end{aligned} \tag{14}$$

(c) **Induction Step** : From Figure 26 and the above hypothesis given in equation (14), we have

$$\begin{aligned} PE_k : \quad x_{k+1} &= X_k + 1 \\ &= k+1 \\ y_{k+1} &= y_{k+1}^1 + y_{k+1}^2 \\ &= (y_k^1 - 1) + (y_k^2 + \delta) \\ &= (N-k-1) + (\sum_{y=N-k}^{N-1} f(x,y) + \delta) \\ &= (N-k-1) + \sum_{y=N-k}^{N-1} f(x,y) + \sum_{y=N-k-1}^{N-k} f(x,y) \\ &= (N-(k+1)) + \sum_{y=N-(k+1)}^{N-1} f(x,y) \end{aligned}$$

We have assumed the theorem to hold for PE_k; we have shown it to hold for PE_1 (the basis); and we have proved under these conditions, that it holds for PE_{k+1}. Therefore, it holds for all PEs.

□

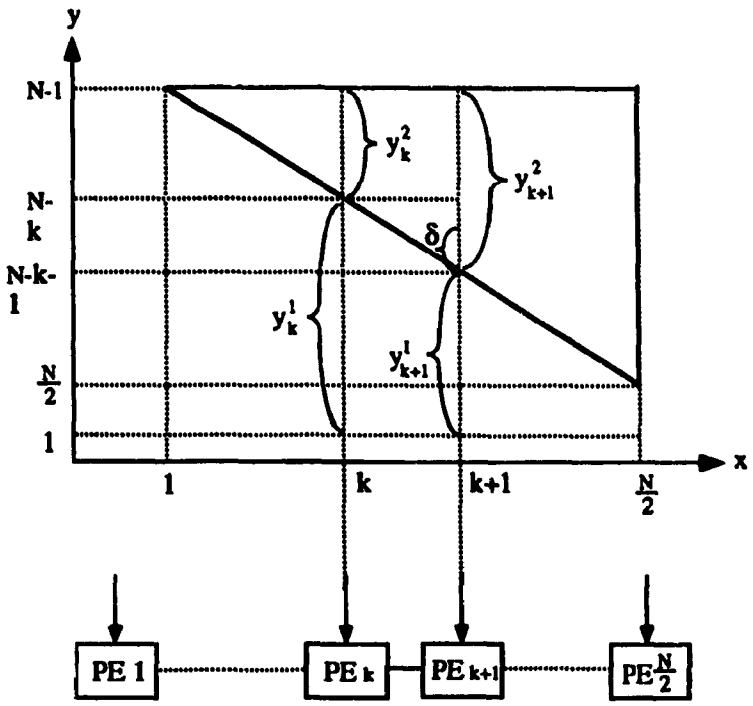


Figure 26: The proof of Theorem 5.1.1.1

5.1.2 Systolic Architecture Design for Sub-DDRPCTs 'D'-'B'

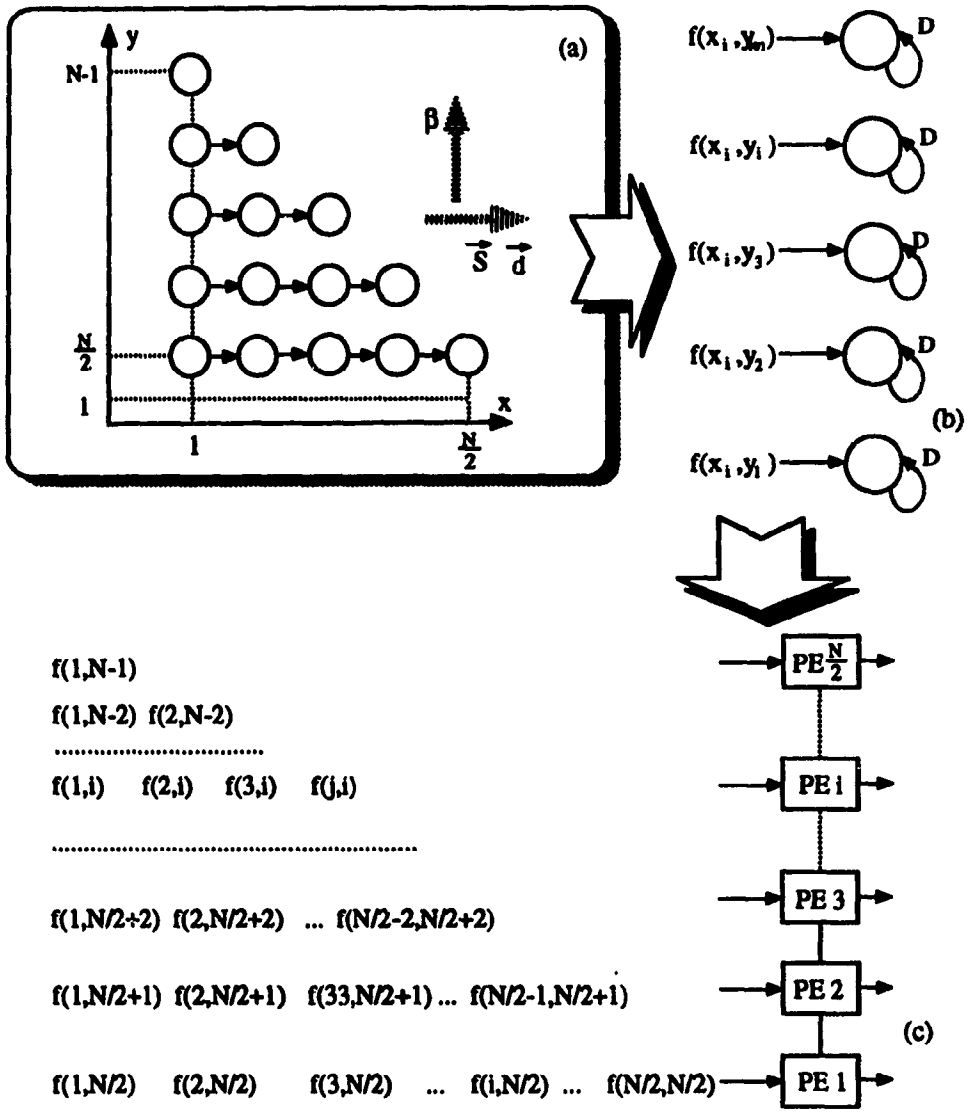
A DG for sub-DDRPCT in sub-region 'D' is shown in Figure 27(a). This is also a local DG, all its nodes are dependent on the neighboring nodes only, it has no global communications in it.

To achieve an efficient VLSI architecture, the DG is mapped into an SFG by an algebraic transformation. Consider the DG of sub-DDRPCT 'D' as shown in Figure 27(a), a *default schedule* is used, the schedule vector is parallel to the projection direction. The projection vector $\vec{d}$ and schedule vector $\vec{s}$ are computed as follows:

$$\vec{d} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad \vec{s} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The processor basis β is obtained orthogonal to $\vec{d}$:

$$\beta^T \vec{d} = 0 \implies \beta = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



(a) DG, (b) SFG, and (c) VLSI architecture

Figure 27: Algorithm-array mapping in subdiagonal-diagonal RCPT'D'.

1 Node mapping

$$\beta^T \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}^T \begin{bmatrix} x \\ y \end{bmatrix} = [y]$$

2 Arc mapping

In the x direction:

$$\begin{bmatrix} D(\vec{e}_x) \\ \vec{e}_x \end{bmatrix} = \begin{bmatrix} \vec{s}^T \\ \beta^T \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

In the y direction:

$$\begin{bmatrix} D(\vec{e}_y) \\ \vec{e}_y \end{bmatrix} = \begin{bmatrix} \vec{s}^T \\ \beta^T \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

3 Input/output mapping

Input:

$$\begin{bmatrix} T_i(\gamma) \\ \Im_i \end{bmatrix} = \begin{bmatrix} \vec{s}^T \\ \beta^T \end{bmatrix} \begin{bmatrix} 1 \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ y \end{bmatrix}$$

Output:

$$\begin{bmatrix} T_o(\gamma) \\ \Im_o \end{bmatrix} = \begin{bmatrix} \vec{s}^T \\ \beta^T \end{bmatrix} \begin{bmatrix} 1 \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ y \end{bmatrix}$$

Once the above mappings have finished, a SFG for sub-DDRPCT in sub-region 'D' can be obtained as shown in Figure 27(b).

From this SFG, a systolic architecture with linear array can be designed as depicted in Figure 27(c). The input data flow $f(x, y)$ is shown in this figure, the output $P_D(x, y)$ of the array is

$$P_D(x, y) = (P_D^1, P_D^2, ..., P_D^i, ..., P_D^{\frac{N}{2}})$$

According to the following theorem, it is clear that the coordinates of the partial contour points of the sub-DDRPCT pattern can be represented by such outputs.

Theorem 5.1.2.1 Let $P_D^i(x,y)$, $i = 1,2,...,\frac{N}{2}$ be outputs of the processors of the systolic array shown in the above. The values of coordinates of the points in the partial contour of sub-DDRPCT 'D' can be described by these outputs, such that

$$P_D^i : P(x,y) \begin{cases} x = (\frac{N}{2} - (i-1)) - \sum_{x=1}^{\frac{N}{2}-i-1} f(x,y) \\ y = \frac{N}{2} + (i-1) \end{cases} \tag{15}$$

Proof: An induction method has been used to prove this theorem.

(a) **Basis** : The case k=1. Immediately the output of processing element PE_1 can be calculated by

$$PE_1 : x_1 = \frac{N}{2} - \sum_{x=1}^{\frac{N}{2}} f(x, \frac{N}{2}+1), \qquad y_1 = \frac{N}{2}$$

(b) **Induction Hypothesis** : Assume in case of k, the output of processing element PE_k is presented by

$$\begin{aligned} PE_k : \quad x_k &= \underbrace{(\frac{N}{2} - (k-1))}_{x_k^1} - \underbrace{\sum_{x=1}^{\frac{N}{2}-(k-1)} f(x,y)}_{x_k^2} \\ y_k &= \frac{N}{2} + (K-1) \end{aligned} \tag{16}$$

(c) **Induction Step** : From Figure 28 and the above hypothesis given in equation (16), we have

$$\begin{aligned} PE_{k+1} : \quad x_{k+1} &= x_{k+1}^1 - x_{k+1}^2 \\ &= (x_k^1 - 1) - (x_k^2 - \delta) \\ &= ((\frac{N}{2} - (k-1)) - 1) - (\sum_{x=1}^{\frac{N}{2}-(k-1)} f(x,y) - \delta) \\ &= (\frac{N}{2} - k + 1 - 1) - (\sum_{x=1}^{\frac{N}{2}-(k-1)} f(x,y) - \sum_{x=\frac{N}{2}-k}^{\frac{N}{2}-(k-1)} f(x,y)) \\ &= (\frac{N}{2} - ((k-1)+1)) - (\sum_{x=1}^{\frac{N}{2}-((k-1)+1)} f(x,y)) \\ y_{k+1} &= y_k + 1 \end{aligned}$$

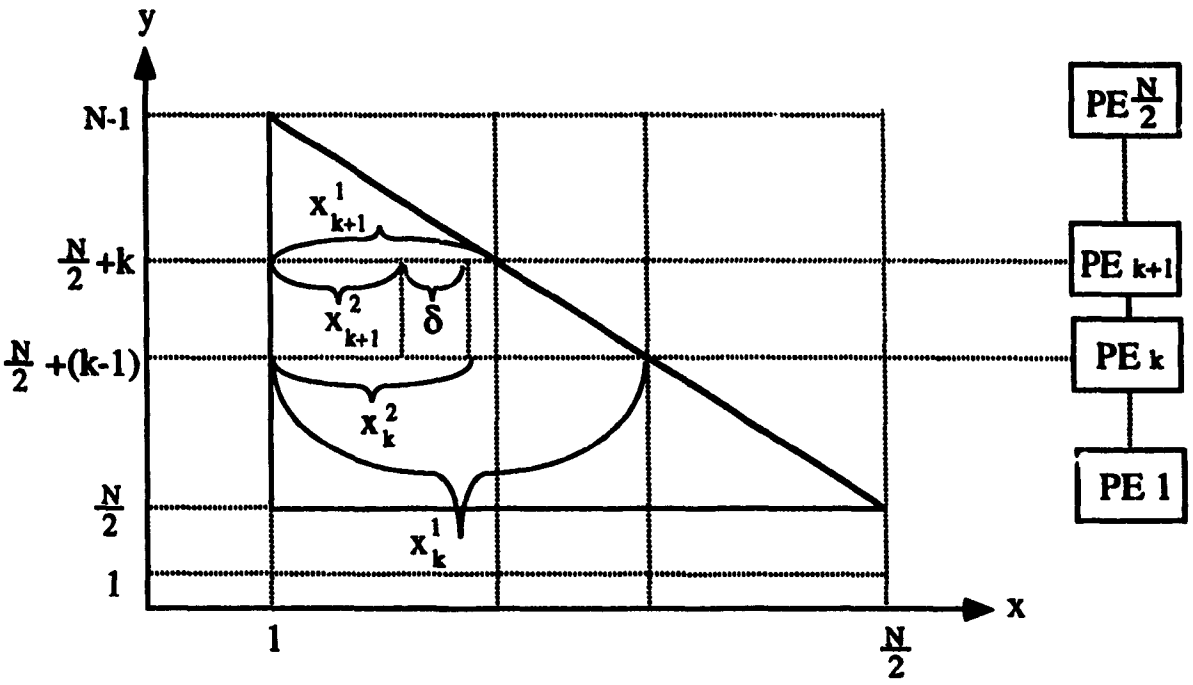


Figure 28: The proof of Theorem 5.1.2.1

$$= (\frac{N}{2} + (K-1)) + 1$$
$$= \frac{N}{2} + ((K-1)+1)$$

□

Similarly, for Sub-DDRPCTs in sub-regions 'E'-'B', the following theorem can be presented:

Theorem 5.1.2.2 Let $P_E^i(x,y)$ - $P_B^i(x,y)$ be a point in the partial contour of sub-regions 'E' - 'B' respectively. The values of these coordinates are described below:

$$P_E^i : P(x,y) \begin{cases} x = i - \sum_{x=1}^{i} f(x,y) \\ y = i \end{cases}$$

$$P_F^i : P(x,y) \begin{cases} x = i \\ y = i - \sum_{y=1}^{i} f(i,y) \end{cases}$$

$$P_G^i : P(x,y) \begin{cases} x = \frac{N}{2} + (i-1) \\ y = (\frac{N}{2} - i + 1) - \sum_{y=1}^{\frac{N}{2}-i-1} f(x,y) \end{cases} \quad (17)$$

$$P_H^i : P(x,y) \begin{cases} x = (N-i) - \sum_{x=N-i}^{N-1} f(x,y) \\ y = i \end{cases}$$

$$P_A^i : P(x,y) \begin{cases} x = (\frac{N}{2} + i - 1) - \sum_{x=\frac{N}{2}+i-1}^{N-1} f(x,y) \\ y = \frac{N}{2} + (i-1) \end{cases}$$

$$P_B^i : P(x,y) \begin{cases} x = \frac{N}{2} + (i-1) \\ y = (\frac{N}{2} + i - 1) - \sum_{y=\frac{N}{2}+i-1}^{N-1} f(x,y) \end{cases}$$

Proof: The theorem can be proved similarly as the proofs of the above theorems. An induction proof can be used.

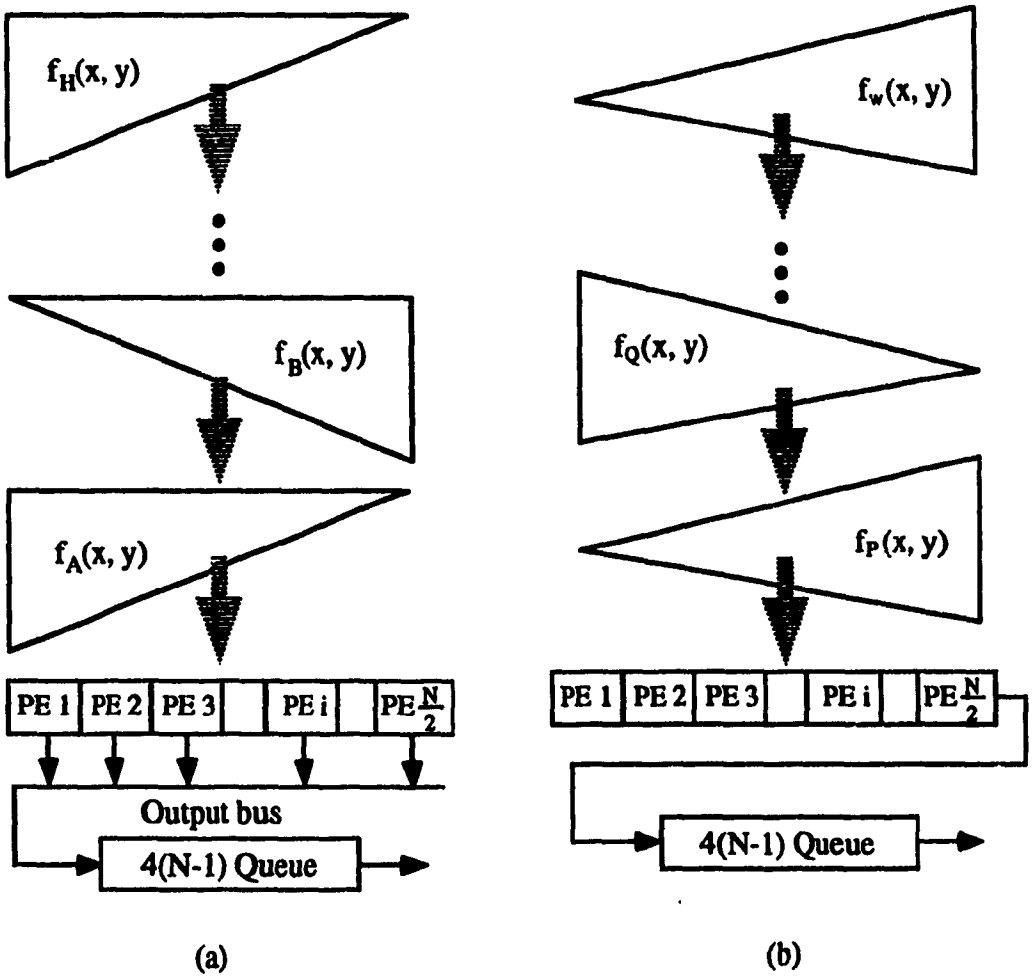
□

5.1.3 Contour Chain for DDRPCT Patterns

A chain of the contour for a DDRPCT pattern can be obtained by connecting the eight outputs of all sub-regions A - H. As described in Figure 29(a), eight groups of input data, $f_A(x,y)$, $f_B(x,y)$, $f_C(x,y)$, $f_D(x,y)$, $f_E(x,y)$, $f_F(x,y)$, $f_G(x,y)$ and $f_H(x,y)$, are sequentially read in the processor array, see Figure 29(a). For each group, the output data is obtained at different PEs. They may either be output sequentially or fetched in parallel through an output bus. In Figure 29(a), the outputs of the DDRPCT are fetched in parallel through an output bus. They are sent to a queue with size of 4(N-1) in the following order:

$$\begin{array}{lll} P_A^{\frac{N}{2}} \sim P_A^1 & \Longrightarrow & P_B^{\frac{N}{2}} \sim P_B^1 \quad \Longrightarrow \\ P_C^{\frac{N}{2}} \sim P_C^1 & \Longrightarrow & P_D^1 \sim P_D^{\frac{N}{2}} \quad \Longrightarrow \\ P_E^1 \sim P_E^{\frac{N}{2}} & \Longrightarrow & P_F^1 \sim P_F^{\frac{N}{2}} \quad \Longrightarrow \\ P_G^1 \sim G_E^{\frac{N}{2}} & \Longrightarrow & P_H^{\frac{N}{2}} \sim P_H^1. \end{array} \tag{18}$$

In the sequel, the chain of the contour for a DDRPCT pattern can be obtained from the queue.



(a) for diagonal-diagonal RPCT
(b) for horizontal-vertical RPCT

F'gure 29: Contour chain of a RPCT pattern obtained from a queue.

5.2 Systolic Array for HVRPCT

Resembling the DDRPCT, an algorithm-array mapping has been used to design a systolic architecture for HVRPCT. In this section, the sub-HVRPCT 'R' is exploited as an example to describe how a systolic array can be designed to implement the HVRPCT.

5.2.1 Algorithm-Array Mapping in Sub-HVRPCT 'R'

A DG for sub-HVRPCT in sub-region 'R' is shown in Figure 30(a). This is also a local DG, all nodes of it are dependent on the neighboring nodes located on the lines with 315° only. As well it has no global communications in it.

To achieve an efficient VLSI architecture, the DG is mapped into an SFG by an algebraic transformation. Consider the DG of sub-DDRPCT 'R' as shown in Figure 30(a), a *default schedule* is used, the schedule vector is parallel to the projection direction. the projection vector $\vec{d}$ and schedule vector $\vec{s}$ are computed as follows:

$$\vec{d} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad \vec{s} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

The processor basis β is obtained orthogonal to $\vec{d}$:

$$\beta^T \vec{d} = 0 \implies \beta = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

1 Node mapping

$$\beta^T \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}^T \begin{bmatrix} x \\ y \end{bmatrix} = [x + y]$$

2 Arc mapping

$$\begin{bmatrix} D(\vec{e}) \\ \vec{e} \end{bmatrix} = \begin{bmatrix} \vec{s}^T \\ \beta^T \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

3 Input/output mapping

Input:

$$\begin{bmatrix} T_i(\gamma) \\ \Im_i \end{bmatrix} = \begin{bmatrix} \vec{s}^T \\ \beta^T \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x-y \\ x+y \end{bmatrix}$$

Output:

$$\begin{bmatrix} T_o(\gamma) \\ \Im_o \end{bmatrix} = \begin{bmatrix} \vec{s}^T \\ \beta^T \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x-y \\ x+y \end{bmatrix}$$

As shown in Figure 30(b), an SFG for sub-HVRPCT in sub-region 'R' can be obtained by assigning the nodes in the index space of the DG to particular nodes of the SFG, and scheduling the order in which these nodes are to be computed.

From this SFG, a systolic array can be designed and described in Figure 30(c), where N=10 is taken as an example to simplify presentation. According to the following theorem, it is clear that the coordinates of the partial contour points of the sub-DDRPCT pattern can be represented by the output $P_R(x,y)$ of the array, i.e.

$$P_R(x,y) = (P_R^1, P_R^2, ..., P_R^i, ..., P_R^{\frac{N}{2}})$$

Theorem 5.2.1.1 Let $P_R^i(x,y)$, $i = 1,2,..., \frac{N}{2}$ be outputs of the processors of the systolic array shown in the above. The values of coordinates of the points in the partial contour of sub-DDRPCT 'R' can be described by these outputs, such that

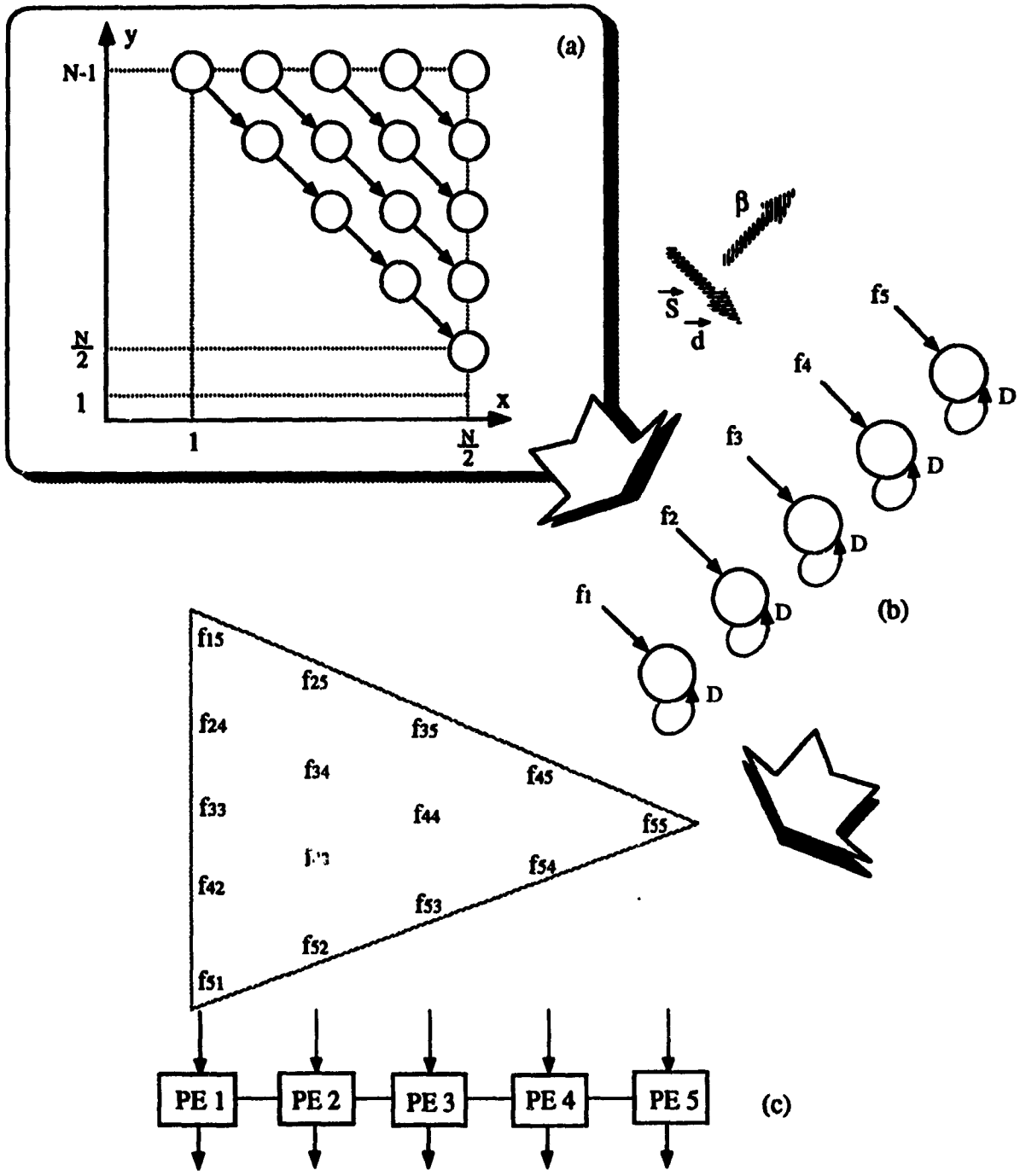
$$P_R^i : P(x,y) \begin{cases} x = (\frac{N}{2}) - \sum_{x+y=N+i-1} f(x,y) \\ y = \frac{N}{2} + (i-1) + \sum_{x+y=N+i-1} f(x,y) \end{cases} \quad (19)$$

Proof: The proof of this theorem resembles the proofs of the above theorems. An induction proof can be used. To save space, it is omitted here.

□

5.2.2 Systolic Architecture Design for Sub-HVRPCTs 'S'-'Q'

Similarly, for Sub-HVRPCTs in sub-regions 'S'-'Q', the following theorem can be presented:



(a) DG, (b) SFG, and (c) VLSI architecture

Figure 30: Algorithm-array mapping in subdiagonal-diagonal RCPT'R'.

Theorem 5.2.2.1 Let $P_S^i(x,y)$ - $P_Q^i(x,y)$ be the points in the partial contours of sub-regions 'S' - 'Q' respectively. The values of these coordinates are described below:

$$
\begin{aligned}
P_S^i &: P(x,y)\begin{cases} x = i - \sum_{x+y=\frac{N}{2}+i} f(x,y) \\ y = \frac{N}{2} + \sum_{x+y=\frac{N}{2}+i} f(x,y) \end{cases} \\
P_T^i &: P(x,y)\begin{cases} x = i + \sum_{x-y=\frac{N}{2}-i} f(x,y) \\ y = \frac{N}{2} - \sum_{x-y=\frac{N}{2}-i} f(x,y) \end{cases} \\
P_U^i &: P(x,y)\begin{cases} x = \frac{N}{2} - \sum_{x-y=i-1} f(x,y) \\ y = i - \sum_{x-y=i-1} f(x,y) \end{cases} \\
P_V^i &: P(x,y)\begin{cases} x = \frac{N}{2} + \sum_{x+y=\frac{N}{2}+i} f(x,y) \\ y = i - \sum_{x+y=\frac{N}{2}+i} f(x,y) \end{cases} \\
P_W^i &: P(x,y)\begin{cases} x = (\frac{N}{2} + i - 1) + \sum_{x+y=N+i-1} f(x,y) \\ y = \frac{N}{2} - \sum_{x+y=N+i-1} f(x,y) \end{cases} \\
P_P^i &: P(x,y)\begin{cases} x = (\frac{N}{2} + i - 1) + \sum_{x-y=i-1} f(x,y) \\ y = \frac{N}{2}) + \sum_{x-y=i-1} f(x,y) \end{cases} \\
P_Q^i &: P(x,y)\begin{cases} x = \frac{N}{2} + \sum_{y-x=i-1} f(x,y) \\ y = (\frac{N}{2} + i - 1) + \sum_{y-x=i-1} f(x,y) \end{cases}
\end{aligned}
\tag{20}
$$

Proof: The theorem can be proved similarly as the proofs of the above theorems. □

5.2.3 Contour Chain for HVRPCT Patterns

A chain of the contour for a HVRPCT pattern can be obtained by connecting the eight outputs of all sub-regions P - W. Eight groups of input data, $f_P(x,y)$, $f_Q(x,y)$, $f_R(x,y)$, $f_S(x,y)$, $f_T(x,y)$, $f_U(x,y)$, $f_V(x,y)$ and $f_W(x,y)$, are sequentially read in the processor array, see Figure 29(b). For each group, the output data are obtained at different PEs. They may either be output sequentially or fetched in parallel through an output bus. In Figure 29(b), the outputs of the HVRPCT are sent sequentially to a queue with size of 4(N-1) in the following order:

$$P_P^{\frac{N}{2}} \sim P_P^1 \implies P_Q^{\frac{N}{2}} \sim P_Q^1 \implies$$

$$\begin{aligned} P_R^{\frac{N}{2}} \sim P_R^1 \quad &\Longrightarrow \quad P_S^1 \sim P_S^{\frac{N}{2}} \quad \Longrightarrow \\ P_T^1 \sim P_T^{\frac{N}{2}} \quad &\Longrightarrow \quad P_U^1 \sim P_U^{\frac{N}{2}} \quad \Longrightarrow \\ P_V^1 \sim G_V^{\frac{N}{2}} \quad &\Longrightarrow \quad P_W^{\frac{N}{2}} \sim P_W^1. \end{aligned} \tag{21}$$

In the sequel, the chain of the contour for a HVRPCT pattern can be obtained from the queue.

5.3 Characteristics of RPCT Approach

The proposed RPCT approach has two major merits: (1) Patterns transformed by the RPCT contain only one outer contour which can greatly simplify multi-contour processing problems; (2) RPCT combines projection operation and contour tracing, hence it can speed up computation.

5.3.1 Characteristics of RPCT Patterns

A very important characteristic of the *DDRPCT* and *HVRPCT* patterns can be presented in the following theorem, which plays a major role in solving the multiple contour problem.

Theorem 5.3.1.1 Either *DDRPCT* pattern or *HVRPCT* pattern has unique outer contour with it.

Proof: First, we prove the case of the *DDRPCT* pattern. A *DDRPCT* pattern P_{DD} consists of 8 sub-patterns P_A - P_H, such that $P_{DD} = P_A \cup P_B \cup P_C \cup P_D \cup P_E \cup P_F \cup P_G \cup P_H$. Furthermore, these eight sub-patterns are generated by eight sub-projections. Each sub-projection is made along a horizontal or vertical line. For instance, the sub-pattern P_C is made by $C(270°, x) = \sum_{y=N-x}^{N} f(x, y)$, for $x = 0, 1, 2, ..., \frac{N}{2}$. It reveals that all pixels in area A are projected onto a diagonal along the horizontal line which produces no interval in the horizontal direction. Thus no hole exists in sub-pattern P_C. Similarly, sub-patterns P_B, P_F and P_G do not contain any hole in them, since sub-projections in areas B, F and G produce no intervals in the horizontal direction. Meanwhile, no intervals along the vertical direction occur in areas A, D, E and H

after projecting all pixels along the vertical. The whole $DDRPCT$ pattern does not contain any hole because all the sub-patterns do not contain any holes. Subsequently, the $DDRPCT$ pattern has only one outer contour.

The case of the $HVRPCT$ pattern resembles that of $DDRPCT$.

□

Although contour analysis has been amenable to shape recognition, it still has a great difficulty dealing with the *multiple-contour* problem which occurs in two cases [113]:

- Compound patterns including unconnected patterns and patterns with isolated noise;
- Patterns with internal contours.

However both cases can be handled according to the characteristic stated in the above theorem. The pattern transformed from the $DDRPCT$ and $HVRPCT$ contains only one outer contour. It makes sense to simplify the process.

5.3.2 Time Complexity

The time complexity of the proposed parallel RPCT algorithm can be computed as follows :

$$T = T_{DD} + T_{HV} + T_C$$

where,

- T_{DD} stands for time units needed for DDRPCT A - H. Pixels in each sub-area, $p_i \in R_j$, have different projection paths. The longest one must be considered when the time complexity is computed.
- T_{HV} denotes time units needed for HVRPCT P - W. The longest path in each projection must also be considered in HVRPCT.
- T_C indicates time units needed for outputting contour chain from queue.

$$
\begin{aligned}
T_{DD} &= \max_{p_i \in A}(\text{time for sub-DDRPCT A}) + \max_{p_i \in B}(\text{time for sub-DDRPCT B}) \\
&+ \max_{p_i \in C}(\text{time for sub-DDRPCT C}) + \max_{p_i \in D}(\text{time for sub-DDRPCT D}) \\
&+ \max_{p_i \in E}(\text{time for sub-DDRPCT E}) + \max_{p_i \in F}(\text{time for sub-DDRPCT F}) \\
&+ \max_{p_i \in G}(\text{time for sub-DDRPCT G}) + \max_{p_i \in H}(\text{time for sub-DDRPCT H}) \\
&= 8 \times \frac{N}{2} = 4N. \\
T_{HV} &= \max_{p_i \in P}(\text{time for sub-HVRPCT P}) + \max_{p_i \in Q}(\text{time for sub-DDRPCT Q}) \\
&+ \max_{p_i \in R}(\text{time for sub-HVRPCT R}) + \max_{p_i \in S}(\text{time for sub-DDRPCT S}) \\
&+ \max_{p_i \in T}(\text{time for sub-HVRPCT T}) + \max_{p_i \in U}(\text{time for sub-HVRPCT U}) \\
&+ \max_{p_i \in V}(\text{time for sub-HCRPCT V}) + \max_{p_i \in W}(\text{time for sub-HVRPCT W}) \\
&= 8 \times \frac{N}{2} = 4N. \\
T_C &= \text{time for moving contour chain of DDRPCT from queue} \\
&+ \text{time for moving contour chain of HVRPCT from queue} \\
&= 2 \times 4N = 8N.
\end{aligned}
$$

Therefore

$$T = 4N + 4N + 8N = O(N)$$

Chapter 6

Evaluation and Experiment

In this chapter, an important theorem will be presented to estimate the quality of features extracted. It will be used to evaluate the DDRPT and HVRPT comparing with other approaches, such as, Cell histogram (CH), Crossing counts and Shading (CCS), and Elastic partitioning (EP) [111, 126, 127].

Suppose a target pattern set contains m classes, each class has L different intrinsic distortions, and each distortion has q samples. The distribution model of patterns with intrinsic distortions has been developed in [126] which can be presented below:

$$P(x) = \sum_{k=1}^{L} P_k p_k(x) \tag{22}$$

where $k = 1, 2, ..., L$ stand for different intrinsic distortions, P_k and p_k are the a priori probability and distribution density of the k-th distortion. Let $m^{(n)}$ be the n-th moment of the overall distribution, $m_k^{(n)}$ be the n-th moment of the k-th distortion, and suppose the *expectation* of the k-th distortion is μ_k,

Theorem 6.1 If a feature extraction method is resistant to intrinsic distortion, the ratio

$$(\frac{m_k^{(n)}}{\mu_k})^{\frac{1}{n}} \tag{23}$$

and

$$(\frac{m_k^{(n)}}{m^{(n)}})^{\frac{1}{n}} \tag{24}$$

should be small.

Proof: More generally, (22) can be rewritten by

$$F(x) = \sum_{k=1}^{L} P_k F_k(x) \tag{25}$$

where F_k and $F_k(x)$ are the *feature distribution functions* of the overall pattern class and k-th intrinsic distortion respectively. Without loss of generality, we suppose $P(x)$ or $F(x)$ has been centralized such that

$$\int_{-\infty}^{\infty} xP(x)dx = 0. \tag{26}$$

The *characteristic function* $\Phi(t)$ of $F(x)$ can be represented as [128]

$$\Phi(t) = \int_{-\infty}^{\infty} e^{itx} dF(x). \tag{27}$$

Similarly, the *characteristic function* $\Phi_k(t)$ of $F_k(x)$ can be described as

$$\Phi_k(t) = \int_{-\infty}^{\infty} e^{itx} dF_k(x), \qquad k = 1, 2, ..., L \tag{28}$$

in general, $F_k(x)$'s have not been centralized.

By a straight forward computation, the *n-th moment* of the overall distribution denoted by $m^{(n)}$ can be derived from the following formula:

$$\begin{aligned} m^{(n)} &= \int_{-\infty}^{\infty} x^j dF(x) \\ &= \frac{1}{i^n} \frac{\partial}{\partial t} \Phi(t)|_{i=0} \\ &= \frac{1}{i^n} \frac{\partial^k}{\partial t} \sum_{k=1}^{L} P_k \Phi_k(t) \end{aligned}$$

Let $a_k^{(n)}$ be the n-th moment of the k-th distortion, which is not necessarily a central moment, we have

$$m^{(n)} = \sum_{k=1}^{L} P_k a_k^{(n)}, \quad n = 1, 2, ... \tag{29}$$

Suppose the *expectation* of the k-th distortion is μ_k,

$$\mu_k = \int x dF_k(x)$$

then

$$\begin{aligned} a_k^{(n)} &= \int (x-\mu_k+\mu_k)^n dF_k(x) \\ &= \sum_{s=0}^{n} \frac{n!}{(n-s)!s!} \int (x-\mu_k)\mu_k^{n-s} dF_k(x) \\ &= \sum_{s=0}^{n} \frac{n!}{(n-s)!s!} m_k^{(s)} \mu_k^{n-s} \end{aligned} \tag{30}$$

where $m_k^{(s)}$ denotes the s-th moment of a sample within the k-th distortion. To facilitate classification, the quantity of $m_k^{(s)}$ should be relatively much smaller than μ_k, i.e. the smaller the ratio of $(\frac{m_k^{(s)}}{\mu_k})^{\frac{1}{s}}$ is, the better the features are. It is obvious in Fig. 31.

From (30) we can conclude that if μ_k is much larger than $m_k^{(s)}$'s then

$$a_k^{(n)} \cong \mu_k \tag{31}$$

by combining equations (29) and (31), we can further conclude that $m^{(n)}$ is much larger than $m_k^{(n)}$'s. This implies that the overall moment is much larger than those due to various intrinsic distortions, i.e. the smaller the ratio of $(\frac{m_k^{(s)}}{m^{(s)}})^{\frac{1}{s}}$ is, the better the features are. It can also be justified in Figure 31.

□

In feature space, relations between the 1-moment of $m_k^{(1)}$, μ_k and $a_k^{(1)}$ can be illustrated graphically in Figure 31. All samples within an intrinsic distortion construct a Normal-distribution region with center c_i. All these regions within a class create a larger region with center c_{fi}. The moment $m_k^{(1)}$ denotes the distance from a sample to the center c_k in the k-distortion. The moment $a_k^{(1)}$ indicates the distance from this sample to the center c_{fi} while μ_k stands for that between these two centers.

The values obtained from (23) and (24) have been used to assess the features extracted by the TRP's. Four methods, Cell histogram (CH) [127], Crossing counts and Shading (CCS) [126], Elastic partitioning (EP) [111] and TRP, have been compared by means of the n-th moment in this study. The values of 2-nd - 5-th moments are computed for these methods with their results illustrated in Tables 4 - 7.

In Tables 4 - 7, the smaller the values, the better the features. It is obvious that the features extracted from the proposed method are much better than that of others.

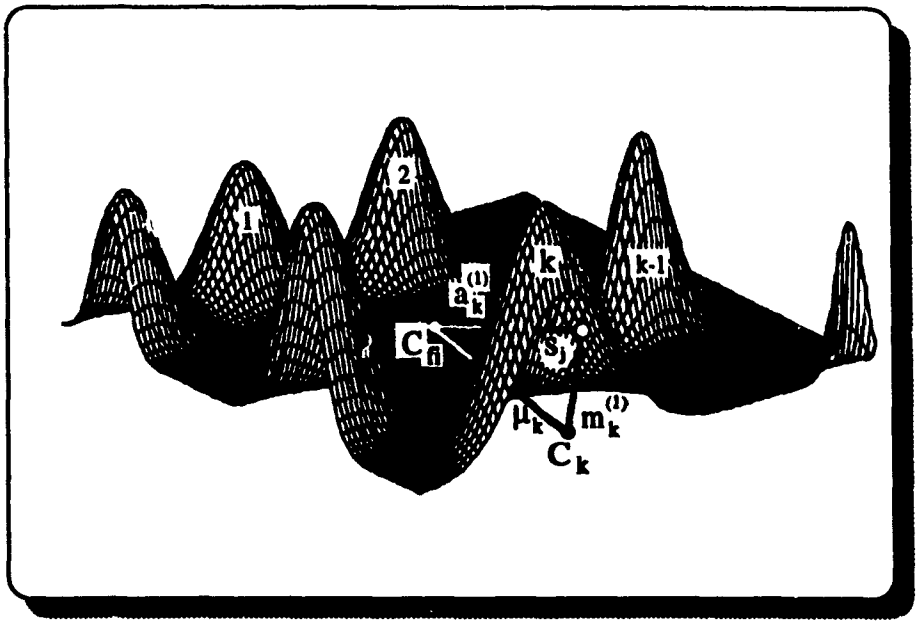


Figure 31: Relations among three moments.

Table 4: Values of the 2-nd moment for different methods

	CH method	CCS method	EP method	TRP method
$(\frac{m_k^{(2)}}{\mu_k})^{\frac{1}{2}}$	8.74	2.95	2.01	1.81
$(\frac{m_k^{(2)}}{m_0^{(2)}})^{\frac{1}{2}}$	7.21	2.14	1.63	1.13

Table 5: Values of the 3-rd moment for different methods

	CH method	CCS method	EP method	TRP method
$(\frac{m_k^{(3)}}{\mu_k})^{\frac{1}{3}}$	2.49	0.82	0.67	0.56
$(\frac{m_k^{(3)}}{m_0^{(3)}})^{\frac{1}{3}}$	2.67	0.68	0.54	0.36

Table 6: Values of the 4-th moment for different methods

	CH method	CCS method	EP method	TRP method
$(\frac{m_k^{(s)}}{\mu_k})^{\frac{1}{s}}$	6.62	2.11	1.86	1.19
$(\frac{m_k^{(s)}}{m_{(s)}})^{\frac{1}{s}}$	5.64	1.73	1.45	0.97

Table 7: Values of the 5-th moment for different methods

	CH method	CCS method	EP method	TRP method
$(\frac{m_k^{(s)}}{\mu_k})^{\frac{1}{s}}$	1.86	0.68	0.48	0.19
$(\frac{m_k^{(s)}}{m_{(s)}})^{\frac{1}{s}}$	0.50	0.47	0.27	0.08

The features from the DDRPT and HVRPT have been used to recognize a large set of characters which contain a lot of compound patterns. The features have been sent to the ISOETRP classifier [127]. In this way, the global decision is accomplished by a series of local decisions made at various levels of the tree. It is suitable for recognizing large data sets such as Chinese characters. When the number of pattern classes, n, in the given problem is very large, the decision making time needed by a simple stage classifier has an order O(n). This can be minimized to O(log n) in the tree classifier, where each pattern class corresponds to one or a few terminal nodes and the height of the tree is in the order O(log n). This is especially significant in the recognition of large character sets. The primary principle of this classifier will be described below.

In conventional clustering, each object is simply a point in the data space or feature space. While in the tree classifier design, a cluster, or a subgroup, contains a number of pattern classes instead of simple point. Consider that each pattern class occupies a region in the feature space. This region can be determined according to the distribution density of the pattern class and must be large enough so that the probability for the patterns of this class falling in this region is near 1.

At the first stage, the root node is the only internal node to be developed, which

contains all pattern classes. ISOETRP is applied to each undeveloped internal node to produce two types of nodes: (a) a terminal if it contains only one pattern class, (b) an undeveloped internal node if it contains more than one class. The process continues until all internal nodes have been developed.

This process is rather local due to the reason that when an internal node is being developed, other nodes are not considered. The tree search in the recognition phase is also rather local, which causes error accumulation. In order to eliminate the locality of the tree, we have to improve the tree search as well as the tree design. The solution is tree global training.

(1) Number of features used at each node.

Suppose each pattern class has the covariance matrix $\sum_i$ and mean vector M_i. It exists in the ellipsoidal region. The pattern space Ω_i for the i-th class can be described as

$$\Omega_i = \{X|(X - M_i)^t \sum_i^{-1}(X - M_i) < \alpha\} \qquad i = 1, 2, ..., n$$

where X is a pattern which belongs to this class, α is a threshold which is as large as possible under the following condition:

$$\Omega_i \cap \Omega_j = \phi, \quad i \neq j, \qquad i, j = 1, 2, ..., n.$$

P is probability which depends on α as well as the pattern class distribution. When it is Gaussian,

$$P = P(\alpha, m)$$

where m is the dimension of the feature space. P values corresponding to different m's and α's are calculated and listed in Table 8.

The larger α is, the nearer P is to 1. While for the same value of α, the larger m value is, the smaller P is.

In the present study, eight features were used at each node and α was detected as 3.8. Usually, the more features are used at each node, the easier it is to separate pattern class groups in generating new nodes. A higher recognition rate of the tree can also be expected, but the tree may take up more memory.

Table 8: The $p(\alpha, m)$ values

α	2.50	2.75	3.00	3.20	3.40	3.60	3.80	4.00
m=1	0.9878	0.9941	0.9975	0.9983	0.9991	0.9996	0.9999	0.9999
m=2	0.9559	0.9769	0.9888	0.9991	0.9968	0.9982	0.9990	0.9996
m=3	0.8996	0.9439	0.9709	0.9830	0.9907	0.9951	0.9975	0.9987
m=4	0.8185	0.9807	0.9390	0.9637	0.9789	0.9886	0.9939	0.9968
m=5	0.7174	0.8180	0.8905	0.9310	0.9584	0.9760	0.9868	0.9931
m=6	0.6039	0.7278	0.8262	0.8849	0.9276	0.9561	0.9747	0.9864
m=7	0.4888	0.6269	0.7469	0.8248	0.8832	0.9270	0.9560	0.9745
m=8	0.3705	0.5225	0.6574	0.7510	0.8278	0.8870	0.9287	0.9577
m=9	0.2849	0.4209	0.5630	0.6688	0.7699	0.8299	0.8943	0.9320
m=10	0.2058	0.3276	0.4682	0.5789	0.6851	0.7752	0.8466	0.9000

(2) Collect possible coming classes.

After the tree has been designed by clustering, global training is applied to the tree. The first step is to do fuzzy logic search or any heuristic search. For each pattern class, all terminals it may reach are found and output. The second step is for any specific terminal to collect all classes, which may reach this terminal [127].

(3) Similarity defining.

Suppose for terminal Z, all possible coming classes have been collected. Suppose the classes other than Z are

$$Y_1, Y_2, ..., Y_k.$$

For each feature f , compute the minimum Mahalanobis distance

$$d_f = \min_{i=1,2,...,k} \left\{ \frac{|Z - Y_i|^2}{\sigma_Z \sigma_{Y_i}} \right\}$$

where Z and Y_i represent the corresponding class centers, and σ_Z, σ_{Y_i} the corresponding class deviations. The m features with smallest minimum Mahalanobis distances are selected as similarity measures at this terminal.

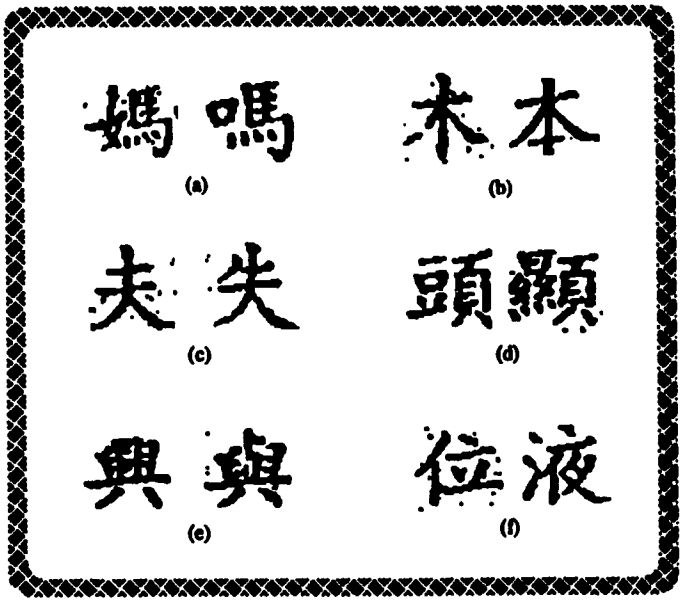
After the similarity for each terminal has been defined, the tree can be modified with these new similarity measures to complete the global training process.

Encouraging results have been achieved in an experiment of recognizing Chinese characters mixed with English characters and numerals. The recognition rates are

Table 9: Recognition results

number of characters	1000	2000	3000
error rate	0.08%	0.10%	0.18%
rejection rate	0.12%	0.29%	0.54%
recognition rate	99.80%	99.46%	99.28%

shown in Table 9. Several examples of the ambiguous pattern pairs in this experiment are illustrated in Figure 32, where the patterns in pairs (a) through (e) were misrecognized in the first search but correctly recognized in the second search, the patterns in (f) were misclassified in both searches.



(a) - (e) : the patterns were misrecognized in the first search, but correctly recognized in the second search;
(f) : the patterns were misclassified in both searchs

Figure 32: Ambiguous character pairs.

Chapter 7

Conclusion

Feature extraction is an important technique in pattern recognition, computer vision and image processing. In this thesis, we proposed a new approach called "regional projection transformation" for feature extraction. Two concentration transformation algorithms are discussed: 1) diagonal-diagonal regional projection transformation (DDRPT), 2) horizontal-vertical regional projection transformation (HVRPT). This approach simplifies the recognition of an image by transforming it into a unique outer contour. It is very useful in recognizing multicontour patterns including compound patterns and patterns with holes in them, which are difficult to be processed by current projection techniques.

To further reduce the computation time, we also study the parallelism of proposed regional projection transformation, four parallel algorithms are developed. We also discuss their implementation on one dimensional linear array and two dimensional mesh architectures. These parallel algorithms have time complexity $O(N)$ comparing with $O(N^2)$ in their sequential algorithms.

To evaluate the quality of the features derived by diagonal-diagonal RPT and horizontal-vertical RPT, we conduct experiments and use DDRPT and HVRPT to recognize a large set of characters which contain a lot of compound patterns and make comparison with other approaches. In the following, we make some observations based on our experimental results, and point out several future research directions.

1. Features extracted from the proposed RPT approach are much better than other

methods such as cell histogram (CH), crossing counts and shading (CCS), elastic partitioning (EP).

2. A good performance can be obtained when this approach is used to recognize Chinese characters mixed with English characters and numerals. We can get more than 99% recognition rate.

3. This approach is very effective in recognizing compound patterns which consist of several isolated parts, or isolated noise, it is a general approach and can be used in different applications apart from character recognition.

For the proposed regional projection transformation, further research can be conducted in the following directions.

1. To evaluate the performance of the approach in different applications such as machine parts recognition, finger print recognition, etc.

2. The systolic architectures proposed in this thesis is based on the such assumption that the size of pattern to be recognized is equal or less than the processor array size. When the pattern size is larger than processor array size, we can consider to use algorithm partition technique to further modify the parallel approaches in this thesis, and implement parallel RPT approaches on fixed-size systolic architectures.

References

[1] H. Abelson and P. Andreae. "Information transfer and area-time tradeoffs for VLSI multiplication". *Communication of ACM*, January 1980.

[2] S. Akamatsu and T. Kawatani. "Hierarchical classification of handprinted Kanji using density feature and configuration feature". In *ICTP 83: Int. Conf. on Text Processing*, 1983.

[3] A. Apostolico and A. Negro. "Systolic algorithms for string manipulations". *IEEE Trans. on Computers*, 33(4), April 1984.

[4] G.M. Baudet, F.P. Preparata, and J.E. Vuillemin. "Area-time optimal VLSI circuits for convolution". *IEEE Trans. on Computers*, 32(7), July 1983.

[5] G. Bilardi and F.P. Preparata. "An architecture for bitonic sorting with optimal VLSI performance". *IEEE Trans. on Computers*, 33(7), July 1984.

[6] G. Bongiovanni. "A VLSI network for variable size of FFT's". *IEEE Trans. on Computers*, 32(8), August 1983.

[7] G. Bongiovanni. "Two VLSI structure for the discrete Fourier transform". *IEEE Trans. on Computers*, 32(8), August 1983.

[8] M.A. Bonuccelli, E. Lodi, and L.Pagli. "External sorting in VLSI". *IEEE Trans. on Computers*, 33(10), October 1984.

[9] R.P. Brent and H.T. Kung. "A regular layout for parallel adders". *IEEE Trans. on Computers*, 31(3), March 1982.

[10] R.P. Brent and H.T. Kung. "Systolic VLSI array for polynomial GCD computation". *IEEE Trans. on Computers*, 33(8), August 1984.

[11] R.M. Burger, R.K. Cavin III, W.C. Holton, and L.W. Sumney. "The impact of ICs on computer technology". *Computer*, 17, October 1984.

[12] R. Cappello and K. Steiglitz. "Unifying VLSI array designs with geometric transformations". In *Proc. of 1983 Int'l Conf. on Parallel Processing*, 1983.

[13] T. A. Cass. "A robust parallel implementation of 2-D model-based recognition". In *Proc. Comput. Vision Pattern Recognition Conf.*, pages 879–884, 1988.

[14] K.P. Chan and Y.S. Cheung. "Fuzzy-attribute graph and its application to chinese character recognition". In *ICCPCOL 88: 1988 Int. Conf. on Computer processing of Chinese and Oriental Languages*, 1988.

[15] T.L. Chang and P.D. Fisher. "Programmable systolic arrays". In *Proc. Comcon, spring*, 1982.

[16] C. H. Chen. *"Signal Processing Handbook"*. 1988.

[17] F.H. Cheng, W.H. Hsu, and M.Y. Chen. "Recognition of handwritten Chinese characters by modified Hough transform techniques". *IEEE Trans. Pattern Analysis Mach. Intell.*, 11(4), 1989.

[18] H.D. Cheng and K.S. FU. "VLSI architectures for pattern matching using space-time domain expansion approach". Technical report, School of Electrical Engineering, Purdue University, October 1984.

[19] H.D. Cheng and K.S. Fu. "Algorithm partition and parallel recognition of general context-free languages using fixed-size VLSI architecture". In *Midwest VLSI Workshop*, 1985.

[20] H.D. Cheng and K.S. Fu. "Algorithm partition for a fixed-size VLSI architecture using space-time domain expansion". In *Proc. of the Seventh Symposium on Computer Architecture*, 1985.

[21] H.D. Cheng and K.S. FU. "VLSI architecture for dynamic time-warp recognition of hand-written symbols". Technical report, School of Electrical Engineering, Purdue University, March 1985.

[22] H.D. Cheng, W.C. Lin, and K.S. Fu. "Space-time domain expansion approach to VLSI and its application to hierarchical scene matching". *IEEE Trans. Pattern Anal. Machine Intell.*, 4(3), March 1985.

[23] Y.S. Cheung and C.H. Leung. "Chain-code transform for Chinese character recognition". In *ICCS85: Proc. IEEE 1985 Int. Conf. on Cybern. and Soc.,*, 1985.

[24] Y.T. Chiang and K.S. Fu. "Parallel parsing algorithms and VLSI implementations for syntactic pattern recognition". *IEEE Trans. on Pattern Anal. Machine Intell.*, May 1984.

[25] K.H. Chu and K.S. Fu. "VLSI architectures for high speed recognition of general context-free languages and finite-state languages". In *Proc. 9th Annual Int'l Symp. Comput. Arch.*, April 1982.

[26] H.Y.H. Chuang and Gao He. "Design of problem-size independent systolic array systems". In *Proc. IEEE Int'l Conf. on computer Design: VLSI in Computers*, October 1984.

[27] M.I. Clarke and C.R. Dyer. "Curve detection in VLSI". In K.S. Fu, editor, *VLSI for Pattern Recognition and Image Processing.* Springer-Verlag, 1984.

[28] T.M. Cover and T.J. Wagner. "Topics in statistical pattern recognition". In K.S. Fu, editor, *Digital Pattern Recognition.* Springer-Verlag, 1980.

[29] I. Dinstein, G. M. Landau, and G. Guy. "Parallel (PRAM EREW) algorithms for contour-based 2-D shape recognition". *Pattern Recognition*, 24(10):929–942, 1991.

[30] M.J. Duff and S. Levialdi. *"Languages and Architectures for Image Processing".* Academic press, 1981.

[31] R. Egan. "The effect of VLSI on computer architecture". *Compuer Architecture News*, 10(5), September 1982.

[32] L. Johnsson et al. "A mathematical approach to modeling the flow of data and control in computational networks". In H.T. Kung et al., editor, *VLSI systems and computations*. Computer Science Press, 1981.

[33] A.L. Fisher and H.T. Kung. "Synchronizing large VLSI processor arrays". In *Proc. of the 10th Annual Int'l Symp. on Computer Architecture*, 1983.

[34] A.L. Fisher, H.T. Kung, L.M. Monier, and Y. Dohi. "Architecture of the PSC: a programmable systolic chip". In *Proc. of the 10th Annual Symposium on Computer Architecture*, June 1983.

[35] A.L. Fisher, H.T. Kung, L.M. Monier, and Y. Dohi. "Design of the PSC: a programmable systolic chips". In R. Bryant, editor, *Third Caltech Conf. on Very Large scale Integration*. Computer Science Press, 1983.

[36] M.J. Forster and H.T. Kung. "The design of special-purpose VLSI chips". *Computer*, 13, January 1980.

[37] J.A.B. Fortes. *"Algorithm transformations for parallel processing and VLSI architecture design"*. PhD thesis, University of Southern California, 1983.

[38] J.A.B. Fortes, K.S. Fu, and B.W. Wah. "Systematic approaches to the design of algorithmically specified systolic arrays". In *Proc. Int'l Conf. on Acoustics, Speech and Signal Processing*, 1985.

[39] M.J. Foster and H.T. Kung. "Recognize regular languages with programmable building-blocks". In J.P. Gray, editor, *VLSI 81*. Academic Press, 1981.

[40] K.S. Fu. *"Syntactic Pattern Recognition and Applications"*. Prentice Hall, 1982.

[41] K.S. Fu. "Special computer architectures for image analysis". *Robotica*, 2, 1984.

[42] K.S. Fu. *"VLSI for Pattern Recognition and Image Processing"*. Springer-Verlag, 1984.

[43] K.S. Fu and T. Ichikana. *"Special Computer Architecture for Pattern Processing"*. CRC press, 1982.

[44] K.S. Fu and A. Rosenfeld. "Pattern recognition and image processing". *IEEE Trans. on Computers*, 25, December 1976.

[45] K.S. Fu and A. Rosenfeld. "Pattern Recognition and Computer Vision". *Computer*, 17(10), October 1984.

[46] N. Fujii, H. Sugawara, E. Yamamoto, and C. Ito. "Some results on handprinted Kanji character recognition using the featureextracted from multiple standpoint". *Trans. IECE(Japan)*, (32), 1981.

[47] M.J. Guibas, H.T. Kung, and C.D. Thompson. "Direct VLSI implementation of combinational algorithms". In *Proc. Caltech. Conf. on VLSI*, January 1979.

[48] N. Hagita, S. Naito, and I. Masuda. "Handprinted Kanji characters recognition based on pattern matching method". In *ICTP 83: Int. Conf. on Text Processing*, 1983.

[49] T. Hai, Y. Kabuyama, and E. Yamamoto. "A method of handwritten Kanji character recognition-recognition method by multiiple standpoints and particular shape extraction". *Trans. IECE(Japan)*, (4), 1985.

[50] I.S. Hsu, I.S. Reed, T.K. Truong, and K. Wang et al. "The VLSI implementation of a Reed-Solomon encoder using berlekamp's bit-serial multiplier algorithm". *IEEE Trans. on Computers*, 33(10), October 1984.

[51] K. Hwang and Y.H. Cheng. "Partitioned matrix algorithms for VLSI arithmetic systems". *IEEE Trans. on Computers*, 31(12), December 1982.

[52] K. Hwang and K.S. Fu. "Integrated computer architectures for image processing and database management". *Computer*, 16(1), January 1983.

[53] K. Hwang and S.P. Su. "VLSI architectures for feature extraction and pattern classification". *J. Comput. Graphics and Image Processing*, 24(2), 1983.

[54] J. Ja'Ja' and R.M. Owens. "VLSI sorting with reduced hardware". *IEEE Trans. on Computers*, 33(10), October 1984.

[55] T. Kawatani, T. Tsutsumida, S. Akamatsu, and N. Nakajima. "A new precise recognition method for handprinted Kanji". In *CVPR 85: IEEE Computer Society conf. on Computer Vision and Pattern Recognition*, June 1985.

[56] M. Khail, J. Atallah, and S.R. Kosaraju. "A generalized dictionary machine for VLSI". *IEEE Trans. on Computers*, 34(2), February 1985.

[57] K. Kobayashi, F. Yoda, K. Yamamoto, and H. Nambu. "Recognition of han-printed Kanji character by the stroke matching method". *Pattern Recognition Letter*, 1, 1983.

[58] B. Krishnamurthy. "An improved Min-cut algorithm for partitioning VLSI networks". *IEEE Trans. on Computers*, 33(5), May 1984.

[59] Y.L. Kuang. "Architecture for VLSI design of Reed-solomon decoders". *IEEE Trans. on Computers*, 33(2), February 1984.

[60] R.H. Kuhn. "Transforming algorithms for signal stage and VLSI architectures". In *Proc. Workshop on Interconnection Networks for Parallel and Distributed Processing*, 1980.

[61] H.T. Kung. "Let's design algorithms for VLSI systems". In *Proc. of the Caltech Conf. on VLSI*, January 1979.

[62] H.T. Kung. "Special-purpose devices for signal and image processing: an opportunity in very large scale integration(VLSI)". In *Proc. of the society of Photo-Optical Instrumentation Engineers*, July 1980.

[63] H.T. Kung. "Use of VLSI in algebraic computation: some suggestion". In *Proc. of the 1981 ACM Symposium on Symbolic and Algebraic Computation*, 1981.

[64] H.T. Kung. *"VLSI systems and Computations"*. Computer Science Press, 1981.

[65] H.T. Kung. "Why systolic architectures". *Computer*, 15, January 1982.

[66] H.T. Kung. "Systolic algorithms for the CMU warp processor". In *Proc. Pattern Recognition Conf.*, July 1984.

[67] H.T. Kung and M.S. Lam. "Eafer-scale integration and two-level pipelined implementations of systolic arrays". *Journal of Parallel and distributed Computing*, (1), 1984.

[68] H.T. Kung and P.L. Lehman. "Systolic(VLSI) arrays for relational database operation". In *ACM SIGMOD Int'l Conf. on management of data*, 1980.

[69] H.T. Kung and C.E. Leiserson. "Systolic arrays (for VLSI)". In C. Mead and L. Conway, editors, *Introduction to VLSI systems.* Addision-Wesley, 1980.

[70] H.T. Kung and W.T. Lin. "An algebra for VLSI algorithm design". In *Proc. Conf. on Elliptic Problem Solvers*, 1983.

[71] H.T. Kung and O. Menzilciogla. "Warp: a programmable systolic array processor". In *Real-Time signal Processing VII*, August 1984.

[72] H.T. Kung and R.L. Picard. "One-dimensional systolic arrays for multidimensional convolution and resampling". In K.S. Fu, editor, *VLSI for Pattern Recognition and Image Processing.* Springer-Verlag, 1984.

[73] H.T. Kung and S.W. Song. "A systolic 2-D convolution chip". Technical report, Dept. of Computer Science, Carnagie-Mellon University, TR-CMU-CS-81-110, March 1981.

[74] S.Y. Kung, K.S. Arun, R.J. Gal-Ezea, and D.V.B. Rao. "Wavefront array processor: language, architecture and applications". *IEEE Trans. on Computers*, 31(11), November 1982.

[75] S.Y. Kung, S.N. Jean, S. C. Lo, and P.S. Lewis. "Design methodologies for systolic array: mapping algorithms to architecture". *Signal Processing Handbook (edited by C. H. Chen)*, pages 145–191, 1988.

[76] F.T. Leighton. *"Complexity Issues in VLSI"*. The MIT Press, 1983.

[77] C.E. Leiserson. "Systolic priority queues". In *Proc. Conf. Very Large Scale Integration*, January 1977.

[78] C.E. Leiserson. *"Area-Efficient VLSI Computation"*. ACM-MIT Press, 1983.

[79] G.J. Li and B. W. Wah. "The design of optimal systolic arrays". *IEEE Trans. on Computers*, 34, January 1985.

[80] H.H. Liu and K.S. Fu. "VLSI arrays for minimum-distance classifications". In K.S. Fu, editor, *VLSI for pattern recognition and image processing*. Springer-Verlag, 1984.

[81] P.S. Liu and T.Y. Young. "VLSI array design under constraint of limited I/O bandwidth". *IEEE Trans. on Computers*, 32(12), December 1983.

[82] Y. Liu and T. Kasvand. "A new approach to matchine recognition of chinese characters". In *ICPR 84: Proc. 1982 Int. Conf. on Pattern Recognition*, 1984.

[83] S.Y. Lu and C.L. Lin. "A structural approach to Chinese character recognition". In *ICCS80: Proc. IEEE 1980 Int. Conf. on Cybern. and Soc.*,, 1980.

[84] K. Maeda, Y. Kurosawa, H. Asada, and S. Watanabe. "Handprinted Kanji recognition by pattern matching method". In *ICPR 82: Proc. 1982 Int. Conf. on Pattern Recognition*, 1982.

[85] C. Mead and L. Conway. *"Introduction to VLSI systems"*. Addison-Wesley, 1980.

[86] W.L. MIranker and A. Winkler. "Space-time representations of computational structures". *Computing*, 32, January 1984.

[87] D.I. Moldovan. "On the design of algorithms for VLSI systolic arrays". *Proceedings of the IEEE*, 71(1), January 1983.

[88] D.I. Moldovan and A. Varma. "Design of algorithmically speciallized VLSI devices". In *Proc. 1983 Int'l Conf. on Computer Design*, 1983.

[89] D.I. Moldovan, C. I. Wu, and A.B. Fortes. "Mapping an arbitrarly large QR algorithm into a fixed size VLSI array". In *Proc. of 1984 Int'l Conf. on Parallel Processing*, August 1984.

[90] S. Naito, N. Hagita, and I. Masuda. "Handprinted Kanji recognition by feature matching methods and its application to personal OCR". In *CVPR 83: IEEE Computer Society conf. on Computer Vision and Pattern Recognition*, June 1983.

[91] D. Nath, S.N. Naheshwari, and P.C.P. Bhatt. "Efficient VLSI networks for parallel processing based on orthogonal trees". *IEEE Trans. on Computers*, 32(6), June 1983.

[92] L.M. Ni and A. K. Jain. "A VLSI systolic architecture for pattern clustering". *IEEE Trans. on Pattern Anal. Machine Intell.*, January 1984.

[93] R. Oka. "Handwritten Chinese-Japanese characters recognition by using cellular feature". In *ICPR 82, Proc. 6th Int. Conf. on Pattern Recognition*, 1982.

[94] T.A. Ottmann, A.L. Rosenberg, and L.J. Stockmeyer. "A dictionary machine (for VLSI)". *IEEE Trans. on Computers*, 31(9), September 1982.

[95] T. Pavlidis. *"Algorithm for Graphics and Image Processing"*. Computer Science Press, 1982.

[96] F.P. Preparata. "A mesh-connected area-time optimal VLSI multiplier of large integers". *IEEE Trans. on Computers*, 32(2), February 1983.

[97] K. Preston, Jr., and L. Uhr. *"Multicomputers and Image processing"*. Academic press, 1982.

[98] G. Rabbat. *"Hardware and software concepts in VLSI"*. Van Nostrand Reinhold Company, 1983.

[99] I.V. Ramakrishnan and P.J. Varman. "Modular matrix multiplication on a linear array". *IEEE Trans. on Computers*, 33(11), November 1984.

[100] T.V. Ramakrishnan, D.S. Fussell, and A. Silberschats. "On mapping homogeneous graphs on a linear array-processor model". In *Proc. of 1983 Int'l Conf. on Parallel Processing*, 1983.

[101] C.V. Ramamoorthy and Y.W. Ma. "Impact of VLSI on computer architectures". In N.G. Einspruch, editor, *VLSI electronics*. Academic Press, 1982.

[102] C.V. Ramamoorthy and Y.W. Ma. "Large scale computer systems". In G. Rabbat, editor, *Hardware and Software Concepts*. Van Morstrand Reinhold Company, 1983.

[103] K.H. Rosen. *"Discrete Mathematics and Its Applications"*. Random House, 1988.

[104] A.L. Rosenberg. "Three-dimensional VLSI: a case study". *J. ACM*, 30(3), July 1983.

[105] J.E. Savage. "Area-time tradeoffs for matrix multiplication and related problems in VLSI models". *Journal of Computer and System sciences*, (22), 1981.

[106] I. Sekita, K. Toraichi, R. Mori, and H. Yamada. "Feature extraction of handwritten Japanese characters by spline functions for relaxation matching". *Pattern Recognition*, 1988.

[107] I. Sekita, K. Toraichi, K. Yamamoto, and H. Yamada. "Feature extraction of handwritten Japanese characters by spline functions for relaxation matching". In *ICASSP 87: 1987 Int. Conf. on Acoustics, Speech, and signal Processing*, 1988.

[108] M. Shiono. "A fundamental experiment on handprinted kanji recognition by multidictionary template matching method". *Trans. inf. Proc. Soc. (Japan)*, 27(9), 1980.

[109] A.K. Soman and V.K. Agarwal. "An efficient VLSI dictionary machine". In *The 11th Annual Int'l Symposium on Computer Architecture*, June 1984.

[110] S.W. Song. *"On a high-performance VLSI solution to database problems"*. PhD thesis, Carnagie Mellon University, 1981.

[111] C.Y. Suen, Y. Y. Tang, and Q. R. Wang. "Feature extraction in the recognition of Chinese characters printed in different fonts". In *Proc. 1986 Int. Conf. on Chinese Computing*, pages 136–143, August 1986.

[112] C.Y. Suen and Q.R. Wang. "ISOETRP - an interactive clustering algorithm with new objectives". *Pattern Recognition*, 17(2):211–219, 1984.

[113] Y.Y. Tang, X. Cheng, L. Tao, and C. Y. Suen. "VLSI architecture for parallel concentration-contour approach". In *Proc. 11th Int. Conf. on Pattern Recognition*, pages 151–154, 1992.

[114] F.T. Taylor. "A VLSI residue arithmetic multiplier". *IEEE Trans. on Computers*, 31(6), June 1982.

[115] C.D. Thompson. "Area-time complexity for VLSI". In *Proc. Caltech Conf. on VLSI*, 1979.

[116] C.D. Thompson. "Fourier transforms in VLSI". *IEEE Trans. on Computers*, 32(8), August 1983.

[117] C.D. Thompson. "The VLSI complexity of sorting". *IEEE Trans. on Computers*, 32(12), December 1983.

[118] P.C. Treleaven. "VLSI processor architectures". *Computer*, 15, June 1982.

[119] P.C. Treleaven. *"VLSI architecture"*. Prentice Hall International Inc., 1983.

[120] J. Tsukumo and K. Asai. "Machines printed Chinese and Japanesed charactr recognition method and experoments for reading Japanese pocket book". In *CVPR 86: IEEE Computer Society conf. on Computer Vision and Pattern Recognition*, June 1986.

[121] L. W. Tucker, C. R. Feyman, and D. M. Fritzsche. "Object recognition using the connection machine". In *Proc. Comput. Vision Pattern Recognition Conf.*, pages 871–878, 1988.

[122] J.D. Ullman. *"Computational Aspects of VLSI"*. Computer Science Press, 1984.

[123] R.B. Urquhart. "VLSI architectures for the linear discriminant function classifier". In *IEEE Workshop on Computer architecture for Pattern Analysis and Image Database Management*, October 1983.

[124] J. Vuillemin. "A combinatorial limit to the computing power of VLSI circuits". *IEEE Trans. on Computers*, 32(3), March 1983.

[125] P.P. Wang and R.C. Shian. "Machine recognition of printed Chinese characters via transformation algorithms". *Pattern Recognition*, (5), 1973.

[126] Q.R. Wang, Y.X. Gu, and C.Y. Suen. "A preliminary study on computer recognition of Chinese characters printed in different fonts". In *Proc. 1982 Int. Conf. on Chinese Computing*, pages 344–351, 1982.

[127] Q.R. Wang and C.Y. Suen. "Large tree classifier with heuristic search and global training". *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(1):91–102, 1987.

[128] Z.K. Wang. *" The Fundementals and Applications of Probability Theory"*. Science Publishing House, Beijing, 1971.

[129] D.F. Wann and M.A. Franklin. "Asynchronous and clocked control structures for VLSI based interconnection networks". *IEEE Trans. on Computers*, 32(3), March 1983.

[130] V. Weiser and A. Davis. "A wavefront notation tool for VLSI array design". In H.T. Kung et al., editor, *VLSI systems and computations.* Computer Science Press, 1981.

[131] E.H. Wold and A.M. Despain. "Pipeline and parallel-pipeline FFT processors for VLSI implementations". *IEEE Trans. on Computers*, 33(5), May 1984.

[132] A. Y. Wu, S. K. Bhaskar, and A. Rosenfeld. "Parallel processing of region boundaries". *Pattern Recognition*, 22(2):165–172, 1989.

[133] S. Wu. "A fuzzy pattern recognition method of handprinted characters". *Chin. J. comput.*, 9(5), 1986.

[134] H. Yamada. "Cpntour DP matching method and its application to handprinted Chinese character recognition". In *ICPR 84: Proc. 7th Int. Conf.on Pattern Recognition*, 1984.

[135] E. Yamamoto, N. Fujii, T. Fujita, and J. Tanahashi. "Handwritten Kanji character recognition using the features extracted from multiple standpoints". In *PRIP81: IEEE Computer Society Conf. on Pattern Recognition and Image Processing*, August 1981.

[136] K. Yamamoto, H. Yamada, T. Saito, and I. Sakaga. "Recognition of handprinted characters in the first level of JIS Chinese characters". In *ICPR 86: Proc. 8th Int. Conf. on Pattern Recognition*, 1986.

[137] Y. Yamashita, K. Higuchi, Y. Yamada, and Y. Haga. "Classification of handprinted Kanji character by the structured segment matching method". *Pattern Recognition Letter*, 1, 1983.

[138] M. Yao. "A recognition scheme for restricted written chinese characters". In *ICCPCOL 90: 1990 Int. Conf. on Computer processing of Chinese and Oriental Languages*, 1990.

[139] M. Yasuda. "Extraction of complementary features". *J. AVIRG*, 18, 1983.

[140] M. Yasuda and H. Fujisawa. "An improved correlation method for character recognition". *Syst. Comput. Controls*, 10(2), 1979.

[141] C.S. Yen, I.S. Reed, and T.K. Truong. "Systolic multipliers for finite fields GF(2^m)". *IEEE Trans. on Computers*, 33(4), April 1984.

[142] D.W.L. Yen and A.V. Kulkarni. "Systolic processing and an implementation for signal and image processing". *IEEE Trans. on Computer*, 31(10), October 1982.

[143] T.Y. Yong, P.S. Liu, and Y. Gao. "Reconfiguable VLSI arrays for pattern analysis and image processing". In *IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database management*, 1983.

[144] J. You, H.A. Cohen, and G.R. Xuan. "Computer recognition of chinese characters". In *ICCPCOL 90: 1990 Int. Conf. on Computer processing of Chinese and Oriental Languages*, 1990.

[145] S. Zhang. "A Chinese character recognition system based on pictorial database technique". In *ICPR 82: Proc. 1982 Int. Conf. on Pattern Recognition*, 1982.

[146] X. Zhang and Y. Xia. "The automatic recognition of handprinted chinese characters-a method of extracting an ordered sequence of strokes". *Pattern Recognition Letter*, 1(4), 1983.