DTIC FILE COPY

AFOSR-TR- 88-1283

②

AD-A202 445

Annual Technical Report for

Grant AFOSR-84-0181

NONLINEAR REAL-TIME OPTICAL SIGNAL PROCESSING

A. A. Sawchuk, Principal Investigator
B. K. Jenkins

Signal and Image Processing Institute
University of Southern California
Mail Code 0272
Los Angeles, California 90089-0272
(213)743-5527

Research Sponsored by the

Air Force Office of Scientific Research
Electronics and Materials Science Division
AFOSR/NE
Bolling AFB, Bldg. 410
Washington, D.C. 20332

Approved for public release; distribution unlimited.

Performance Period: 1 July 1987 - 30 June 1988

The United States Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation hereon.

[illegible] OFFICE OF SCIENTIFIC RESEARCH (AFSC)
[illegible] OF TRANSMITTAL TO DTIC
[illegible] report has been reviewed and is
[illegible] public release IAW AFR 190-12
[illegible] is unlimited.
[illegible] KERPER
[illegible] Technical Information Division

DTIC
ELECTE
DEC 1 6 1988
S D
a E

ADA202445

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT The United States Go... and di... p... n... nt Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 88-1283
6a. NAME OF PERFORMING ORGANIZATION Signal and Image Processing Inst. Univ. of Southern Cal. — 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION AFOSR/NE
6c. ADDRESS (City, State, and ZIP Code) MC-0272 Los Angeles, California 90089	7b. ADDRESS (City, State, and ZIP Code) Bldg 410 Bolling AFB DC 20332
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Air Force Office of Scientific Research — 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-84-0181
8c. ADDRESS (City, State, and ZIP Code) Bldg. 410, Bolling AFB Washington, D.C. 20332	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 61102F — PROJECT NO. 2305 — TASK NO. B1 — WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Nonlinear Real-Time Optical Signal Processing

12. PERSONAL AUTHOR(S)
A.A. Sawchuk, B.K. Jenkins

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Annual Technical	FROM 7/1/87 TO 6/30/88	1988 July 1	123

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Optical Computing
Optical Signal Processing

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

During the period 1 July 1987- 30 June 1988, the research under Grant AFOSR-84-0181 has been concerned with binary parallel optical computing architectures with particular attention to cellular logic and symbolic substitution for pattern recognition and numerical operations. Our approach has been to experimentally implement binary optical cellular logic processors and interconnection arrays; define an instruction set and software suited to optical computing systems; and to study generalizations of optical cellular logic processors such as the hypercube and pyramid. Recent accomplishments include the experimental implementation of a 54-gate

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL [illegible]	22b. TELEPHONE (Include Area Code) (202) 767-4931 — 22c. OFFICE SYMBOL NE

DD Form 1473, JUN 86 Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

binary optical cellular logic processor with instruction decoders, input/output, memory and test/branch functions; the completion of a binary image algebra (BIA) description of cellular logic, image analysis and symbolic operations; and the development of binary image algebra algorithms for scale and shift invariant pattern recognition.

Contents

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

DTIC COPY INSPECTED

1 Summary

During the period 1 July 1987 - 30 June 1988, the research under Grant AFOSR-84-0181 has been concerned with binary parallel optical computing architectures with particular attention to cellular logic and symbolic substitution for pattern recognition and numerical operations. Our approach has been to experimentally implement binary optical cellular logic processors and interconnection arrays; define an instruction set and software suited to optical computing systems; and to study generalizations of optical cellular logic processors such as the hypercube and pyramid. Recent accomplishments include the experimental implementation of a 54-gate binary optical cellular logic processor with instruction decoders, input/output, memory and test/branch functions; the completion of a binary image algebra (BIA) description of cellular logic, image analysis and symbolic operations; and the development of binary image algebra algorithms for scale and shift invariant pattern recognition.

2 Research Progress

This section summarizes research progress and accomplishments for the period 1 July 1987 - 30 June 1988 on Grant AFOSR-84-0181 for Nonlinear Real-Time Optical Signal Processing. These results are discussed separately in the sections that follow.

An Image Algebra Representation of Parallel Optical Binary Arithmetic

K. S. Huang, B. K. Jenkins, A. A. Sawchuk

Signal and Image Processing Institute

Department of Electrical Engineering

University of Southern California

Los Angeles, CA 90089-0272

[o†]Supported by the Air Force Office of Scientific Research under grant AFOSR-84-0181, by the office of Naval Research under contract N00014-86-K-0602, and by an IBM graduate fellowship.

[‡]Portions of this paper presented at the 1987 Annual Meeting, Optical Society of America, Rochester, New York, October, 1987 [20].

[*]Submitted to *Applied Optics*.

Abstract

Optical computers can operate on 2-D planes of data in parallel. Boolean logic equations do not provide a complete description of such parallel operations for binary arithmetic. An optical system that operates on planes of data should employ an inherently parallel mathematical description for its arithmetic. The purposes of this paper are: to use binary image algebra to develop parallel numerical computation algorithms, and to describe the execution of these algorithms on a digital optical cellular image processor (DOCIP) architecture. We discuss three basic binary number representations: 1) binary row(or column)-coding; 2) binary stack-coding; and 3) binary symbol-coding for symbolic substitution arithmetic.

1 Introduction

Digital optical systems hold the promise of providing more accuracy, flexibility, and programmability than analog optical systems, at the cost of somewhat lower throughput [1] [2]. To achieve digital optical computing, there are at least three possible logic systems: residue logic [3]-[6], multilevel logic [7]-[10] , and binary logic [11] [12]. Because it is much easier to make reliable two level devices for binary logic and only log_2k of them are needed to to represent k levels, in this paper we will consider only binary parallel optical computing. A digital optical cellular image processor (DOCIP) architecture based on binary image algebra (BIA) has been demonstrated to be very powerful in parallel binary image processing [13]-[16]. This paper will demonstrate that the DOCIP with BIA algebraic techniques can efficiently perform parallel numerical computations also.

Boolean logic equations for binary arithmetic are not well-suited to highly parallel operations on planes of data; they do not reflect the location of data except typically by a memory address. Here, we first seek a software theory for parallel numerical computation algorithms that simultaneously have binary digital efficiency and the advantages of optical parallel processing. We have developed a binary image algebra (BIA) [16], built from only 3 fundamental operations and 5 elementary images, to serve as a complete unified systematic theory for binary parallel image processing. Now, we will show that BIA can be also considered as a spatial logic which is a generalized parallel form of boolean logic with an additional parallel information transfer ability. BIA then becomes a formalism and a general technique for developing and comparing parallel numerical computation algorithms for digital optical computers.

Based on BIA, parallel numerical computation algorithms for the DOCIP machine are developed and compared to that for optical symbolic substitution processors [17]-[19]. Symbolic substitution rules are particular BIA image transformations [20] (section 5). Hence, the comparison of these machines will be in terms of this BIA algebraic language. Three different binary number representations (binary row(or column)-coding, binary stack-coding, and binary symbol-coding of symbolic substitution) for binary arithmetic in the DOCIP machine are developed. Parallel operations of binary addition, subtraction and multiplication are derived by BIA and illustrated as examples. Parallelism is achieved by performing arithmetic operations on many pairs of operands simultaneously. The carries for each pair of operands are essentially propagated serially to keep hardware complexity low [21]. This enables speed-ups close to, and in some cases equal to, linear to be obtained. In this paper we will consider only positive numbers. A suitable digital number representation will easily provide for negative numbers also. For example, two's complement arithmetic can be performed with only minor modifications to the algorithms and programs given in this paper, and with the addition of one more bit (the sign bit) to each operand and result.

Section 2 gives a brief review of BIA and the DOCIP architecture. Section 3 presents binary row(or column)-coded arithmetic: binary addition, binary subtraction, and binary multiplication (including a matrix-constant multiplication and an element-element multiplication). Section 4 presents binary stack-coded arithmetic. Section 5 gives a BIA representation of symbolic substitution and discusses binary symbol-coded arithmetic (symbolic substitution arithmetic). Section 6 gives a comparison for the above different number representations.

2 Binary Image Algebra (BIA) and DOCIP Architecture

2.1 Review of Binary Image Algebra

We give here a very brief summary of BIA. Details are contained in Ref. [16].

A binary digital image is usually defined as a function f mapping each grid point (x, y) of an orthogonal coordinate system onto the set composed of two elements: 1 (white, bright, i.e. image point) and 0 (black, dark, i.e. background point). However, it will be more convenient for our image algebra to use only the set of coordinates of image points ('1's) to specify an image. In BIA, an image is then treated as a set of coordinates of image points (pixels that have value 1). This paper deals with only binary arithmetic; hence, an image point represents a binary bit with value 1, a background point represents a binary bit with value 0, and an image is a finite 2-D bit plane. We list here only those basic definitions and operations which will be referred to latter.

Definition of Binary Image Algebra (BIA)

Binary image algebra is an algebra with an image space S and a family F of operations including 5 elementary images and 3 fundamental operations. Symbolically,

$$BIA = (P(W); \oplus, \cup, ^-, I, A, A^{-1}, B, B^{-1}) \tag{1}$$

where $S = P(W)$ and $F = (\oplus, \cup, ^-, I, A, A^{-1}, B, B^{-1})$. The image space S, the family F of operations, and all other symbols are defined in the following.

1. *The Universal Image* (the bit plane containing all bits with value 1): The universal image is a set $W = \{(x,y) \mid x \in Z_n, y \in Z_n\}$, where $Z_n = \{0, \pm 1, \pm 2, ..., \pm n\}$ and n is a positive integer.

2. *Image Space* (the set of all possible bit planes): The image space is the power set (the set of all subsets) of the universal image, i.e. $S = P(W)$.

3. *Image* (bit plane): A set X is an image if and only if X is an element of the image space S, i.e. X is a subimage of the universal image W.

4. *Image Point* (a bit with value 1): A point (bit) (x,y) is an image point of an image X if and only if (x,y) is an element of the set X.

5. *Image Transformation* (a mapping between bit planes): An image transformation T is a function mapping the image space S into the image space S.

6. *Three Fundamental Operations* (Fig. 1):

   (a) Complement of an image X:

$$\overline{X} = \{(x,y) \mid (x,y) \in W \wedge (x,y) \notin X\} \tag{2}$$

   (b) Union of two images X and R:

$$X \cup R = \{(x,y) \mid (x,y) \in X \vee (x,y) \in R\} \tag{3}$$

   (c) Dilation of two images X and R:

$$X \oplus R = \begin{cases} \{(x_1 + x_2, y_1 + y_2) \in W \mid (x_1, y_1) \in X, (x_2, y_2) \in R\} & (X \neq \phi) \wedge (R \neq \phi) \\ \phi & otherwise \end{cases} \tag{4}$$

Remark: "$\in$" denotes "belongs to", "$\wedge$" denotes "and", "$\vee$" denotes "or", and "ϕ" is the null image having no image point. Note that X usually represents an input image and R is a reference image containing predefined information. We can define other image operations as fundamental operations instead of these three operations. The reason for choosing these three operations is because of their simplicity, simple software design and simple hardware implementation. Dilation can be interpreted as a parallel mathematical formalism of the pattern substitution step in symbolic substitution (section 5).

7. *Five Elementary Images*: There are 5 elementary images:

   (a) $I = \{(0,0)\}$ — consisting of an image point at the origin

   (b) $A = \{(1,0)\}$ — consisting of an image point right of the origin

   (c) $A^{-1} = \{(-1,0)\}$ — consisting of an image point left of the origin

   (d) $B = \{(0,1)\}$ — consisting of an image point above the origin

   (e) $B^{-1} = \{(0,-1)\}$ — consisting of an image point below the origin

In fact, these 5 elementary images could be reduced to 4 elementary images, because $I = A \oplus A^{-1} = B \oplus B^{-1}$. Any (reference) image can be represented as

$$X = \bigcup_{(i,j)\in X} A^i B^j \tag{5}$$

where $A^i B^j \equiv A^i \oplus B^j$,

$A^i \equiv \underbrace{A \oplus A \oplus \ldots \oplus A}_{i} = \{(i,0)\}$ if $i > 0$,

$A^i \equiv \underbrace{A^{-1} \oplus A^{-1} \oplus \ldots \oplus A^{-1}}_{-i} = \{(i,0)\}$ if $i < 0$,

$A^0 \equiv A \oplus A^{-1} = I$.

8. *Reflected Image*: Given an image R, its refected image is defined as

$$\check{R} = \{(-x,-y) \mid (x,y) \in R\}. \tag{6}$$

9. *Some Standard Derived Operations*:

(a) Difference of X by R (Fig. 2(a)):

$$X/R = \{(x,y) \mid (x,y) \in X \wedge (x,y) \notin R\} = X \cap \overline{R} = \overline{\overline{X} \cup R} \tag{7}$$

Remark: $\overline{X} = W/X$ where W is the universal image.

(b) Intersection of two images X and R (Fig. 2(b)):

$$X \cap R = \{(x,y) \mid (x,y) \in X \wedge (x,y) \in R\} = \overline{\overline{X} \cup \overline{R}} \tag{8}$$

Remark: $X \cup R = \overline{\overline{X} \cap \overline{R}}$.

(c) Erosion of an image X by a reference image R (Fig. 2(c)):

$$X \ominus R = \overline{\overline{X} \oplus \check{R}} \tag{9}$$

where $\check{R}$ is defined above.

Remark: $X \oplus R = \overline{\overline{X} \ominus \check{R}}$. The erosion of an image X by a reference image R can be thought as the complement of the dilation of the background by the reflection of the reference image R. In general, the erosion of a non-null image X by a non-null reference image R decreases the size of regions, increases the size of holes, eliminates regions, and breaks bridges in X.

(d) Symmetric difference of two images (Fig. 2(d)):

$$X \bigtriangleup R = (X/R) \cup (R/X) = \overline{\overline{X} \cup R} \cup \overline{\overline{R} \cup X} \tag{10}$$

Remark: The symmetric difference is a commutative operation, and is its own inverse.

(e) Hit or miss transform $\circledast$ of an image X by an image pair $R = (R_1, R_2)$ (Fig. 2(e)):

$$X \circledast R = (X \ominus R_1) \cap (\overline{X} \ominus R_2) = \overline{(\overline{X} \oplus \check{R}_1) \cup (X \oplus \check{R}_2)} \tag{11}$$

Remark: The hit or miss transform of an image X by a reference image pair $R = (R_1, R_2)$ formally describes the pattern recognition step in symbolic substitution (section 5); and it is used to match the shape (or template) defined by the reference image pair R where R_1 defines the foreground of the shape and R_2 defines the background of the shape. The conditions are that the foreground X must match R_1 (i.e. $X \ominus R_1$), while simultaneously the background $\overline{X}$ matches R_2 (i.e. $\overline{X} \ominus R_2$). Note the similarity of the symmetric difference (parallel bit-wise comparison) and the hit or miss transform (parallel shape or symbol recognition).

The important results of BIA are: (1) any image transformation can be implemented by the three fundamental operations with appropriate reference images; (2) any reference image can be generated from the elementary images by using the three fundamental operations; and (3) BIA provides an efficient representation for many parallel image processing algorithms (e.g. shape and size verifications [16]). Here we will demonstrate that BIA is also a fundamental tool for parallel numerical computation.

2.2 Review of DOCIP Architecture

We have designed a class of the digital optical cellular image processors (DOCIPs) for effectively implementing BIA [13]-[15]. Here we only summarize their major characteristics. Details are given in [14] [15]. To map BIA into the DOCIP architecture in a transparent way, we first define the DOCIP algebraicly:

Definition of Cellular Automata

A cellular automaton is an algebra $A = (S; F, N_c)$ where S is the state space which is a set of states, F is a family of transition functions, and N_c is the neighborhood configuration.

Constraints on a cellular automaton for Implementing BIA:

1. $S \supset P(W)$
2. $F \supset \{\oplus, \cup, ^{-}\}$
3. $N_c \supset I \cup A \cup A^{-1} \cup B \cup B^{-1}$ or $N_c \supset A \cup A^{-1} \cup B \cup B^{-1}$

where "$\supset$" means "contains".

Thus, in terms of cellular automata, the DOCIPs have to satisfy the above constraints for realizing BIA. For storing input images and temporary results in a more flexible way, the DOCIPs utilize three memory modules and all share the same algebraic structure (except the neighborhood configuration):

$$DOCIP = (P(W \times W \times W); \oplus, \cup, ^{-}, N_c) \tag{12}$$

where "$\times$" denotes cross product and N_c can be one of the following 4 types:

1. DOCIP-array4: each cell connects with its four nearest neighbors and itself, i.e.

$$N_{array4} = I \cup A \cup A^{-1} \cup B \cup B^{-1}. \tag{13}$$

2. DOCIP-array8: each cell connects with its eight nearest neighbors and itself, i.e.

$$N_{array8} = \bigcup_{i,j=-1}^{1} A^i B^j. \tag{14}$$

3. DOCIP-hypercube4: each cell connects with those cells in the 4 directions at distances $1, 2, 4, 8, ..., 2^k$ from itself, i.e.

$$N_{hypercube4} = \bigcup_{i=0,\pm 1,\pm 2,...,\pm 2^k} (A^i \cup B^i) \tag{15}$$

where k is sufficiently large for the connections to traverse the entire array of cells.

4. DOCIP-hypercube8: each cell connects with those cells in the 8 directions at distances $1, 2, 4, 8, ..., 2^k$ from itself, i.e.

$$N_{hypercube8} = \bigcup_{i=0,\pm 1,\pm 2,...,\pm 2^k} (A^i \cup B^i \cup A^i B^i \cup A^i B^{-i}) \tag{16}$$

From the above algebraic description, the DOCIPs have the same algebraic structure and differ only in their neighborhood configurations N_c. Thus, they share the same architecture shown in Fig. 3, but have different configurations of the reference images E_i depending on the optical interconnection network which defines the neighborhood. In practical applications, a *larger* reference image R can be generated from a set of *smaller* reference image(s) E_i by a "sequential dilation". If it is possible to decompose R into a sequence $R = E_1 \oplus E_2 \oplus ... \oplus E_k$, then

$$X \oplus R = (...((X \oplus E_1) \oplus E_2) \oplus ... \oplus E_k). \tag{17}$$

This decomposition may not exist, in which case R can always be decomposed as $R = R_1 \cup R_2 \cup ... \cup R_k$, and then

$$X \oplus R = (X \oplus R_1) \cup (X \oplus R_2) \cup ... \cup (X \oplus R_k) \tag{18}$$

where each R_j can be decomposed into smaller reference images E_i [14] [22].

Basically, the proposed DOCIP shown in Fig. 3 is a cellular SIMD machine and consists of an array of cells or processing elements (PEs) under the supervision of a control unit. The control unit includes a clock, a program counter, a test and branch module for feedback control, and an instruction decoder for storing instructions and decoding them to supervise cells. The array of cells includes a 1×3 line destination selector, where each line is N^2 bits wide, three $N \times N \times 1$ bit memories for storing images, a memory selector, and a dilation unit. It operates as follows: (1) a binary image ($N \times N$ matrix) is input into the destination selector and then stored in any memory (or set of memories) as the instruction specifies; (2) after one to three images have been stored, these images and their complements are piped into the next stage, which forms the union of any combination of images (specified by the instruction); (3) the result is sent to a dilation unit where the reference image specified by the instruction is used to control the type of dilation; (4) finally, the dilated image can be output, tested for program control, or fed back to step (1) as the instruction specifies.

The DOCIP machine (Fig. 3) has one instruction; it implements the three fundamental operations of BIA along with fetch and store [22]. This design uses the parallelism of optics to simultaneously execute instructions involving all N^2 picture elements. Each instruction takes one complete cycle to execute. Note that the DOCIP machine can perform a dilation by any reference image R that is a subset of the neighborhood configuration, N_c, in a single clock cycle.

The entire system can be realized by an optical gate array with optical 3-D interconnections [11] [12] [23]. Fig. 4 describes an optical implementation concept for the DOCIP-hypercube. The DOCIP has very low cell hardware complexity to maximize parallelism, yet enough cell sophistication to permit the machine to execute useful programs. The use of optical interconnections permits a cellular hypercube topology to be implemented without paying a large penalty in chip area (the cellular hypercube interconnections are space-invariant which implies relatively low hologram complexity); it also enables images to be input to and output from the machine in parallel.

3 Binary row(or column)-coded Arithmetic

Binary addition of two k-bit numbers yields at most $k+1$ bits, and binary multiplication of two k-bit numbers yields at most $2k$ bits. In this paper. we assume that all input numbers are padded with enough zeroes to avoid the possibility of overflow. This also guarantees that the different operands in the image will be treated separately. A binary row-coded number is encoded in a part of a row of an image. Although the word lengths of numbers do not need to be equal, we assume in this discussion that an image (bit plane) with $N \times N$ bits contains N^2/k numbers of k-bit length as a simple illustration (Fig. 5). In this section, we describe parallel addition, subtraction and multiplication by BIA expressions and their programs on the DOCIP machine.

3.1 Addition of Binary row-coded Numbers

Consider an image X (e.g. Fig. 6(a)) composed of N^2/k numbers $x_i, i = 1, 2, ..., N^2/k$, an image R (e.g. Fig. 6(b)) composed of N^2/k numbers $r_i, i = 1, 2, ..., N^2/k$, and the output of the addition $S = X + R$ (Fig. 6(c)). To realize this addition in parallel by means of BIA, we first consider the serial (carray-propagate) addition of 2 binary numbers $s_i = x_i + r_i$. The first step of serial addition is to add the least significant bits, say $x_{i(o)}$ and $r_{i(o)}$. The boolean logic equations for adding the two least significant bits (half-adder) are

- sum bit: $s_{i(o)} = x_{i(o)}$ XOR $r_{i(o)}$,
- carry bit: $c_{i(o)} = x_{i(o)}$ AND $r_{i(o)}$.

Now, applying the corresponding parallel operations of XOR and AND, i.e. the symmetrical difference $\triangle$ and intersection $\cap$, and shifting the set of carry bits by a dilation $\oplus$, we can implement parallel addition by the following recursive equations:

1. Define the initial states of images of sum bits and carry bits (called sum-bit image and carry-bit image) at time t_o as :

$$S(t_0) = X, \;\; C(t_0) = R. \tag{19}$$

2. The recursive relation between the states of the sum-bit image and carry-bit image at two adjacent time intervals is then:

$$S(t_{i+1}) = S(t_i) \bigtriangleup C(t_i) = \overline{\overline{S(t_i)} \cup C(t_i) \cup \overline{S(t_i) \cup \overline{C(t_i)}}} \tag{20}$$

$$C(t_{i+1}) = (S(t_i) \cap C(t_i)) \oplus A^{-1} = \overline{\overline{S(t_i)} \cup \overline{C(t_i)}} \oplus A^{-1} \tag{21}$$

where $i = 0, 1, 2, ..., k+1$, and the elementary image A^{-1} is used to shift the carry-bit image one bit to the left for the next iteration.

3. After a maximum of $k+1$ iterations, the sum-bit image is the result and the carry-bit image is the null image ϕ:

$$S(t_{k+1}) = X + R, \quad C(t_{k+1}) = \phi. \tag{22}$$

This procedure is illustrated in Fig. 6(d). The result of parallel addition of binary numbers with a maximum k-bit word size is obtained after $k+1$ iterations. This algorithm can be implemented in the DOCIP architecture by the program (instructions) given below. M_1, M_2, and M_3 represent the three $N \times N$-bit memories. "$X \rightarrow M_1$" denotes "store X into memory M_1". Each numbered line represents a single DOCIP machine instruction for one value of i. Comments are in parentheses.

- Assume start with X in M_1 $(= S(t_0))$ and R in M_2 $(= C(t_0))$.

- First to k^{th} iterations:

  1. $\overline{M_1} \cup \overline{M_2} \rightarrow M_3$ $(= \overline{S(t_i)} \cup \overline{C(t_i)})$
  2. $\overline{M_1} \cup M_2 \rightarrow M_1$ $(= \overline{S(t_i)} \cup C(t_i))$
  3. $\overline{M_1} \cup \overline{M_2} \cup \overline{M_3} \rightarrow M_2$ $(= S(t_i) \cup \overline{C(t_i)})$
  4. $\overline{M_1} \cup \overline{M_2} \rightarrow M_1$ $(= S(t_{i+1}))$
  5. $\overline{M_3} \oplus A^{-1} \rightarrow M_2$ $(= C(t_{i+1}))$

  where $i = 0, 1, 2, ..., k-1$.

- $(k+1)^{th}$ iteration:

  1. $\overline{M_1} \cup M_2 \rightarrow M_3$ $(= \overline{S(t_k)} \cup C(t_k))$

2. $M_1 \cup \overline{M_2} \rightarrow M_1$ ($= S(t_k) \cup \overline{C(t_k)}$)

3 $\overline{M_1} \cup \overline{M_3} \rightarrow$ Out ($= S(t_{k+1}) = X + R$).

The total number of clock cycles for the execution of this program on the DOCIP machine is

$$t(k) \leq 5k + 3 = O(k)$$

which is independent of the number of words being added.

In fact, BIA can be used to devise a parallel form of a conditional-sum adder or carry-lookahead adder for further extracting additional parallelism, and the execution time of this addition can be reduced to $O(log_2 k)$. Obviously, there exists a tradeoff between execution time and hardware complexity. This paper concentrates only on some simple algorithms .

3.2 Subtraction of Binary row-coded Numbers

Let the output of the parallel subtraction be $D = X - R$ (e.g. Fig. 7(a)-(c)). To realize it, we first consider the serial binary subtraction of 2 binary numbers $d_i = x_i - r_i$. The procedure in the least significant bits $x_{i(o)}$ and $r_{i(o)}$ of binary subtraction generates a difference bit $d_{i(o)}$ and a borrow bit $b_{i(o)}$. The boolean logic equations for subtracting the two least significant bits (half-subtractor) are

- difference bit: $s_{i(o)} = x_{i(o)}$ XOR $r_{i(o)}$,
- borrow bit: $c_{i(o)} = \overline{x}_{i(o)}$ AND $r_{i(o)}$.

Now, applying the corresponding parallel operations, and shifting the set of borrow bits by a dilation' $\oplus$, we can implement the parallel subtraction as follows:

1. Define the initial states of images of difference bits and borrow bits (called difference-bit image and borrow-bit image) at time t_o as :

$$D(t_0) = X, \;\; B(t_0) = R. \tag{23}$$

2. The recursive relation between the states of the difference-bit image and borrow-bit image at two adjacent time intervals is:

$$D(t_{i+1}) = D(t_i) \bigtriangleup B(t_i) = \overline{\overline{D(t_i)} \cup B(t_i)} \cup \overline{D(t_i) \cup \overline{B(t_i)}} \tag{24}$$

$$B(t_{i+1}) = (\overline{D(t_i)} \cap B(t_i)) \oplus A^{-1} = \overline{D(t_i) \cup \overline{B(t_i)}} \oplus A^{-1} \tag{25}$$

where $i = 0, 1, 2, ..., k+1$, and the elementary image A^{-1} is used to shift the borrow-bit image one bit to the left for the next iteration.

3. After a maximum of $k+1$ iterations, the difference-bit image is the result and the borrow-bit image becomes the null image ϕ:

$$D(t_{k+1}) = X - R, \quad B(t_{k+1}) = \phi. \tag{26}$$

This procedure is illustrated in Fig. 7(d). The result of parallel subtraction of binary numbers with a maximum k-bit word size is obtained after $k+1$ iterations. The DOCIP architecture can realize this by the following program (instructions):

- Assume start with X in M_1 ($= D(t_0)$) and R in M_2 ($= B(t_0)$).

- First to k^{th} iterations:

  1. $M_1 \cup \overline{M_2} \rightarrow M_3$ ($= D(t_i) \cup \overline{B(t_i)}$)
  2. $\overline{M_1} \cup M_2 \rightarrow M_1$ ($= \overline{D(t_i)} \cup B(t_i)$)
  3. $\overline{M_1} \cup \overline{M_2} \rightarrow M_1$ ($= D(t_{i+1})$)
  4. $\overline{M_3} \oplus A^{-1} \rightarrow M_2$ ($= B(t_{i+1})$)

  where $i = 0, 1, 2, ..., k-1$.

- $(k+1)^{th}$ iteration:

  1. $M_1 \cup \overline{M_2} \rightarrow M_3$ ($= D(t_k) \cup \overline{B(t_k)}$)
  2. $\overline{M_1} \cup M_2 \rightarrow M_1$ ($= \overline{D(t_k)} \cup B(t_k)$)
  3. $\overline{M_1} \cup \overline{M_2} \rightarrow M_1$ ($= D(t_{k+1}) = X - R$)

The total number of clock cycles in the DOCIP to complete this subtraction process is

$$t(k) \leq 4k + 3 = O(k).$$

3.3 Multiplication of Binary row-coded Numbers

Using the representation illustrated in Fig. 5, we define a parallel (matrix-constant) multiplication of an image set of binary numbers and one single binary number $X \cdot R_r$, and parallel (element-element) multiplication of two image sets of binary numbers $X \times R$.

I. Matrix-Constant Multiplication $X \cdot R_r$

Consider an image X (e.g. Fig. 8(a)) comprising N^2/k numbers $x_i, i = 1, 2, ..., N^2/k$, and a reference image R_r (e.g. Fig. 8(b)) comprising only one single k-bit binary number $r = (r_{(k-1)}r_{(k-2)}...r_{(0)})_2$. The output of the parallel multiplication is $X \cdot R_r$ (Fig. 8(c)). To realize it, we first consider the serial multiplication of two binary numbers that is the sum of the shifted versions of the multiplier or the multiplicand. Then, by applying the corresponding parallel operations and parallel shifting by a dilation $\oplus$, we can implement this parallel multiplication by the equation

$$X \cdot R_r = \sum_{l, \forall r_{(l)}=1} X \oplus A^{-l} \tag{27}$$

where the sum notation $\sum$ refers to a sequence of parallel additions and the parallel addition $+$ is defined in subsection 3.1.

The DOCIP takes $O(k^2)$ clock cycles for implementing this matrix-constant multiplication. Its procedure involves:

1. Generating the term $X \oplus A^{-l}$:

   - The DOCIP-array requires at most $l \leq k - 1 = O(k)$ clock cycles, because

$$\begin{aligned} A^{-l} &= (A^{-1})^l \\ &\equiv \underbrace{A^{-1} \oplus A^{-1} \oplus ... \oplus A^{-1}}_{l} \\ X \oplus A^{-l} &= (...((X \oplus \underbrace{A^{-1}) \oplus A^{-1}) \oplus ... \oplus A^{-1}}_{l}). \end{aligned} \tag{28}$$

- The DOCIP-hypercube requires at most $log_2 l \leq log_2(k-1) = O(log_2 k)$ clock cycles, because we can rewrite l as a binary number $l = (a_{(\lfloor log_2 l \rfloor)} \ldots a_{(1)} a_{(0)})_2$, and we have

$$\begin{aligned} A^{-l} &= \prod_{j=0}^{\lfloor log_2 l \rfloor} A^{-a_{(j)} \cdot 2^j} \\ &\equiv A^{-a_{(0)}} \oplus A^{-a_{(1)} \cdot 2^1} \oplus \ldots \oplus A^{-a_{(\lfloor log_2 l \rfloor)} \cdot 2^{\lfloor log_2 l \rfloor}} \\ X \oplus A^{-l} &= (\ldots((X \oplus A^{-a_{(0)}}) \oplus A^{-a_{(1)} \cdot 2^1}) \oplus \ldots \oplus A^{-a_{(\lfloor log_2 l \rfloor)} \cdot 2^{\lfloor log_2 l \rfloor}}) \end{aligned} \tag{29}$$

  where $\lfloor log_2 l \rfloor$ is the greatest integer less than or equal to $log_2 l$, and each dilation with $A^{-a_{(j)} \cdot 2^j}$ can be implemented in the DOCIP-hypercube in one single clock cycle.

- The total time delay for generating all required $X \oplus A^{-l}$, $0 \leq l \leq k-1$, is bounded by $O(k)$ for both the DOCIP-array and the DOCIP-hypercube. Since

$$X \oplus A^{-l} = (X \oplus A^{-(l-1)}) \oplus A^{-1}, \tag{30}$$

  we can generate the new term $X \oplus A^{-l}$ by simply deriving it from the previous term $X \oplus A^{-(l-1)}$ without starting from the original X. The total generating time is then dominated by the number of terms $X \oplus A^{-l}$ which is at most $O(k)$.

2. Implementing the summation $\sum_{l, \forall r_{(l)}=1} X \oplus A^{-l}$:

   - The DOCIPs require at most $k - 1 = O(k)$ parallel additions to implement this summation, and each parallel addition requires at most $k + 1 = O(k)$ iterations (as shown in subsection 3.1). Since it takes $O(k)$ time for generating all the terms $X \oplus A^{-l}$, the total execution time of the DOCIPs for this matrix-constant multiplication of k-bit binary numbers is

$$O(k) \times O(k) + O(k) = O(k^2).$$

From the example shown in Fig. 8, $R_r = I \cup A^{-2}$ contains only a single number $r = (0101)_2 = 5$, and the DOCIP can implement this matrix-constant multiplication $X \cdot R_r$ as follows:

Assume start with X in M_1 ($= X \oplus I$).

1. $M_1 \oplus A^{-2} \rightarrow M_2$ ($= X \oplus A^{-2}$)

2. The instructions of the parallel addition are performed as shown in subsection 3.1:

$M_1 + M_2 \rightarrow$ Out ($= X \cdot R_r$).

II. Element-Element Multiplication $X \times R$

Consider an image X (e.g. Fig. 9(a)) comprising N^2/k numbers $x_i, i = 1, 2, ..., N^2/k$, and an image R (e.g. Fig. 9(b)) comprising N^2/k numbers $r_i, i = 1, 2, ..., N^2/k$. The output of the element-element parallel multiplication is $X \times R$ (Fig. 9(c)). Because the multiplication of two binary numbers is the sum of the shifted versions of the multiplier or the multiplicand, applying the corresponding parallel operations, we can implement this parallel multiplication by the equation

$$\begin{aligned} X \times R &= \textstyle\sum_{l=0}^{k-1}(X \oplus A^{-l}) \cap ((R \cap (M \oplus A^{-l})) \oplus \cup_{j=0}^{k-l-1} A^{-j}) \\ &= \textstyle\sum_{l=0}^{k-1} \overline{\overline{X \oplus A^{-l}} \cup \overline{\overline{\overline{R} \cup \overline{M \oplus A^{-l}}} \oplus \cup_{j=0}^{k-l-1} A^{-j}}} \end{aligned} \qquad (31)$$

where the mask M (Fig. 9(d)) is used to extract the l^{th} bit (where the 0^{th} bit is least significant and the $(k-1)^{th}$ bit is most significant). The DOCIPs can implement this element-element multiplication by the procedure

1. Generate $X \oplus A^{-l}$ and $\overline{\overline{R} \cup \overline{M \oplus A^{-l}}}$:

   - Using an argument similar to that in subsubsection I above, the DOCIP-array takes $O(k)$ time and the DOCIP-hypercube takes $O(log_2 k)$ time.

2. Generate $\overline{\overline{R} \cup \overline{M \oplus A^{-l}}} \oplus \cup_{j=0}^{k-l-1} A^{-j}$:

   - The DOCIP-array takes $O(k)$ time, because

$$\bigcup_{j=0}^{k-l-1} A^{-j} = (\bigcup_{j=0}^{1} A^{-j})^{k-l-1} \equiv \underbrace{(\bigcup_{j=0}^{1} A^{-j}) \oplus (\bigcup_{j=0}^{1} A^{-j}) \oplus ... \oplus (\bigcup_{j=0}^{1} A^{-j})}_{k-l-1}, \qquad (32)$$

   $l \geq 0$, and each dilation by a term in parentheses executes in one clock cycle.

   - The DOCIP-hypercube takes $O(log_2 k)$ time, since

$$\bigcup_{j=0}^{k-l-1} A^{-j} = \prod_{n=0}^{\lfloor log_2(k-l-1) \rfloor} (\bigcup_{j=0}^{n} A^{-a_{(j)} \cdot 2^j}) \qquad (33)$$

   where $k-l-1 = (a_{(\lfloor log_2(k-l-1) \rfloor)} ... a_{(1)} a_{(0)})_2$, and again each dilation by the term in parentheses executes in one clock cycle.

- It takes $O(k)$ time for the DOCIP-array and $O(log_2k)$ for the DOCIP-hypercube to generate the term $\overline{\overline{(X \oplus A^{-l})} \cup ((\overline{\overline{\overline{R} \cup \overline{(M \oplus A^{-l})}}}) \oplus \cup_{j=0}^{k-l-1} A^{-j})}$.

3. Implementing the summation $\sum_{l=0}^{k-1} \overline{\overline{X \oplus A^{-l}} \cup \overline{\overline{\overline{R} \cup \overline{M \oplus A^{-l}}}} \oplus \cup_{j=0}^{k-l-1} A^{-j}}$:

    - The summation requires at most $(k-1)$ addition operations, and each addition operation takes $O(k)$ time on the DOCIP system. We also require $O(k)$ time for the DOCIP-array and $O(log_2k)$ time for the DOCIP-hypercube to generate each operand of the addition. Thus, for this element-element multiplication of k-bit binary numbers, the total computation time is $O(k^3)$ for the DOCIP-array and $O(k^2log_2k)$ for the DOCIP-hypercube.

Multiplication requires more than three memories. This can be accommodated by either building more memory into the DOCIP machine or by swapping intermediate results into and out of an external memory. In the latter case we assume the external memory can be loaded and unloaded with one image in a single time step. In section 4, binary stack-coded arithmetic also requires more than three memories; we'll make the same assumptions on the use of an external memory.

For binary column-coded arithmetic, a number is encoded in a part of a column of an image as in Fig. 10. All the algorithms derived in this section can be also applied to binary column-coded numbers except that we replace the elementary image A^{-1} by a different elementary image B for shifting the carry-bit image or borrow-bit image in the vertical direction.

4 Binary stack-coded Arithmetic

In this case, a number is encoded in a stack of k image planes with the least significant bit in the first plane, next least significant bit in the second plane, etc. (Fig. 11). We assume all numbers including the results of arithmetic operations can be represented in k bits, so that k images, each with $N \times N$ bits, contain N^2 binary numbers. Here, we describe parallel addition, subtraction and multiplication by BIA expressions.

4.1 Addition of Binary stack-coded Numbers

Using the representation illustrated in Fig. 11, we consider the parallel addition of two sequences of images of binary numbers. Assume a sequence of images $X = (X_{(k-1)}, X_{(k-2)}, \ldots, X_{(0)})$ (e.g. Fig. 12(a)) storing N^2 binary numbers $x_i, i = 1, 2, \ldots, N^2$, and a sequence of images $R = (R_{(k-1)}, R_{(k-2)}, \ldots, R_{(0)})$ (e.g. Fig. 12(b)) storing N^2 numbers $r_i, i = 1, 2, \ldots, N^2$. Then the output of the parallel addition is $X + R = S = (S_{(k)}, S_{(k-1)}, \ldots, S_{(0)})$ as shown in Fig. 12(c). To realize this addition using our three fundamental operations, we implement an array of full adders as described by the equations

1. The least significant bit planes of sum bits and carry bits are given by:

$$S_{(0)} = X_{(0)} \triangle R_{(0)} = \overline{\overline{X_{(0)}} \cup R_{(0)}} \cup \overline{X_{(0)} \cup \overline{R_{(0)}}} \tag{34}$$

$$C_{(1)} = X_{(0)} \cap R_{(0)} = \overline{\overline{X_{(0)}} \cup \overline{R_{(0)}}} \tag{35}$$

2. The recursive relations:

$$\begin{aligned} S_{(i)} &= X_{(i)} \triangle R_{(i)} \triangle C_{(i)} \\ &= \overline{(\overline{X_{(i)}} \cap \overline{R_{(i)}} \cap \overline{C_{(i)}}) \cup (\overline{X_{(i)}} \cap R_{(i)} \cap C_{(i)}) \cup (X_{(i)} \cap \overline{R_{(i)}} \cap C_{(i)}) \cup (X_{(i)} \cap R_{(i)} \cap \overline{C_{(i)}})} \\ &= \overline{\overline{(X_{(i)} \cup R_{(i)} \cup C_{(i)})} \cup \overline{(X_{(i)} \cup \overline{R_{(i)}} \cup \overline{C_{(i)}})} \cup \overline{(\overline{X_{(i)}} \cup R_{(i)} \cup \overline{C_{(i)}})} \cup \overline{(\overline{X_{(i)}} \cup \overline{R_{(i)}} \cup C_{(i)})}} \end{aligned} \tag{36}$$

$$\begin{aligned} C_{(i+1)} &= (X_{(i)} \cap R_{(i)}) \cup (X_{(i)} \cap C_{(i)}) \cup (R_{(i)} \cap C_{(i)}) \\ &= \overline{(\overline{X_{(i)}} \cup \overline{R_{(i)}})} \cup \overline{(\overline{X_{(i)}} \cup \overline{C_{(i)}})} \cup \overline{(\overline{R_{(i)}} \cup \overline{C_{(i)}})} \end{aligned} \tag{37}$$

where $i = 0, 1, 2, \ldots, k-1$.

3. The final solution is:

$$X + R = S = (S_{(k)}, S_{(k-1)}, \ldots, S_{(0)}). \tag{38}$$

where $S_{(k)} = C_{(k)}$ because $X_{(k)} = R_{(k)} = \phi$.

This algorithm can be implemented in the DOCIP architecture by the program (DOCIP instructions):

- Assume start with $X_{(0)}$ stored in M_1 and $R_{(0)}$ stored in M_2.

- Calculate $S_{(0)}$ and $\overline{C_{(1)}}$:

  1. $\overline{M_1} \cup \overline{M_2} \rightarrow M_3$ & Out $(= \overline{C_{(1)}})$
  2. $\overline{M_1} \cup M_2 \rightarrow M_1$ $(= \overline{X_{(0)}} \cup R_{(0)})$
  3. $\overline{M_1} \cup \overline{M_2} \cup \overline{M_3} \rightarrow M_2$ $(= X_{(0)} \cup \overline{R_{(0)}})$
  4. $\overline{M_1} \cup \overline{M_2} \rightarrow$ Out $(= S_{(0)})$

- Calculate $S_{(1)}$ and $C_{(2)}$:

  1. $X_{(1)} \rightarrow M_1$
  2. $\overline{M_1} \cup \overline{M_3} \rightarrow M_2$ $(= \overline{X_{(1)}} \cup C_{(1)})$
  3. $M_1 \cup M_3 \rightarrow M_1$ $(= X_{(1)} \cup \overline{C_{(1)}})$
  4. $\overline{M_1} \cup \overline{M_2} \rightarrow M_1$ $(= X_{(1)} \triangle C_{(1)})$
  5. $R_{(1)} \rightarrow M_2$
  6. $\overline{M_1} \cup M_2 \rightarrow M_3$
  7. $M_1 \cup \overline{M_2} \rightarrow M_2$
  8. $\overline{M_2} \cup \overline{M_3} \rightarrow$ Out $(= S_{(1)})$
  9. $X_{(1)} \rightarrow M_1$
  10. $R_{(1)} \rightarrow M_2$
  11. $\overline{M_1} \cup \overline{M_2} \rightarrow M_3$
  12. $\overline{C_{(1)}} \rightarrow M_1$
  13. $M_1 \cup \overline{M_2} \rightarrow M_2$
  14. $\overline{M_2} \cup \overline{M_3} \rightarrow M_3$
  15. $X_{(1)} \rightarrow M_2$
  16. $M_1 \cup \overline{M_2} \rightarrow M_2$
  17. $\overline{M_2} \cup M_3 \rightarrow M_3$ & Out $(= C_{(2)})$

- Calculate $S_{(2)}$ to $S_{(k-1)}$ and $C_{(3)}$ to $C_{(k)}$:

  Use the same instructions for calculating $S_{(1)}$ and $C_{(2)}$ except that $X_{(1)}$ and $R_{(1)}$ (and $S_{(1)}$ and $C_{(2)}$) are replaced by $X_{(i)}$ and $R_{(i)}$ (and $S_{(i)}$ and $C_{(i+1)}$) in each iteration, and in the beginning of an iteration the memory M_3 stores $C_{(i)}$ instead of $\overline{C_{(1)}}$, $i = 2, 3, .., k$.

The complete execution of this operation in the DOCIP requires

$$t(k) \leq 17(k-1) + 4 = 17k - 13 = O(k).$$

clock cycles. Additional parallelism could be extracted to further reduce the execution time by utilizing carry-lookahead techniques or by optimizing the above program.

4.2 Subtraction of Binary stack-coded Numbers

Let the result of the parallel subtraction be $X - R = D = (D_{(k-1)}, D_{(k-2)}, ..., D_{(0)})$ (e.g. Fig. 12(d)). To realize it using the 3 fundamental operations, we consider a serial full-subtractor. Applying the corresponding parallel operations, we can implement this parallel subtraction by the equations

1. **The least significant bit planes of difference bits and borrow bits:**

$$D_{(0)} = X_{(0)} \bigtriangleup R_{(0)} = \overline{\overline{X_{(0)}} \cup R_{(0)}} \cup \overline{\overline{R_{(0)}} \cup X_{(0)}} \tag{39}$$

$$B_{(1)} = \overline{X_{(0)}} \cap R_{(0)} = \overline{X_{(0)} \cup \overline{R_{(0)}}} \tag{40}$$

2. **The recursive relations:**

$$\begin{aligned} D_{(i)} &= (\overline{X_{(i)}} \cap \overline{R_{(i)}} \cap B_{(i)}) \cup (\overline{X_{(i)}} \cap R_{(i)} \cap \overline{B_{(i)}}) \cup (X_{(i)} \cap \overline{R_{(i)}} \cap \overline{B_{(i)}}) \cup (X_{(i)} \cap R_{(i)} \cap B_{(i)}) \\ &= \overline{(X_{(i)} \cup R_{(i)} \cup \overline{B_{(i)}})} \cup \overline{(X_{(i)} \cup \overline{R_{(i)}} \cup B_{(i)})} \cup \overline{(\overline{X_{(i)}} \cup R_{(i)} \cup B_{(i)})} \cup \overline{(\overline{X_{(i)}} \cup \overline{R_{(i)}} \cup \overline{B_{(i)}})} \end{aligned} \tag{41}$$

$$\begin{aligned} B_{(i+1)} &= (\overline{X_{(i)}} \cap \overline{R_{(i)}} \cap B_{(i)}) \cup (\overline{X_{(i)}} \cap R_{(i)} \cap \overline{B_{(i)}}) \cup (\overline{X_{(i)}} \cap R_{(i)} \cap B_{(i)}) \cup (X_{(i)} \cap R_{(i)} \cap B_{(i)}) \\ &= \overline{(X_{(i)} \cup R_{(i)} \cup \overline{B_{(i)}})} \cup \overline{(X_{(i)} \cup \overline{R_{(i)}} \cup B_{(i)})} \cup \overline{(X_{(i)} \cup \overline{R_{(i)}} \cup \overline{B_{(i)}})} \cup \overline{(\overline{X_{(i)}} \cup \overline{R_{(i)}} \cup \overline{B_{(i)}})} \end{aligned} \tag{42}$$

where $i = 0, 1, 2, ..., k-1$.

3. **The final solution:**

$$X - R = D = (D_{(k-1)}, D_{(k-2)}, ..., D_{(0)}). \tag{43}$$

This algorithm can be implemented in the DOCIP architecture by the program (instructions):

- Assume start with $X_{(0)}$ in M_1 and $R_{(0)}$ in M_2.

- Calculate $D_{(0)}$ and $\overline{B_{(1)}}$:

1. $M_1 \cup \overline{M_2} \rightarrow M_3$ & Out $(= \overline{B_{(1)}})$
2. $\overline{M_1} \cup M_2 \rightarrow M_1$ $(= \overline{X_{(0)}} \cup R_{(0)})$
3. $\overline{M_2} \cup \overline{M_3} \rightarrow$ Out $(= D_{(0)})$

- Calculate $D_{(1)}$ and $B_{(2)}$:

1. $X_{(1)} \rightarrow M_1$
2. $M_1 \cup M_3 \rightarrow M_2$
3. $\overline{M_1} \cup M_3 \rightarrow M_1$
4. $R_{(1)} \rightarrow M_3$
5. $M_2 \cup M_3 \rightarrow M_2$
6. $M_1 \cup M_3 \rightarrow M_3$
7. $\overline{M_2} \cup \overline{M_3} \rightarrow M_2$
8. $R_{(1)} \rightarrow M_3$
9. $M_1 \cup \overline{M_3} \rightarrow M_1$
10. $\overline{M_1} \cup M_2 \rightarrow M_2$
11. $X_{(1)} \rightarrow M_2$
12. $\overline{M_1} \cup M_3 \rightarrow M_3$
13. $B_{(1)} \rightarrow M_1$
14. $M_1 \cup M_3 \rightarrow M_3$
15. $M_2 \cup \overline{M_3} \rightarrow$ Out $(= D_{(1)})$
16. $X_{(1)} \rightarrow M_3$
17. $\overline{M_1} \cup M_3 \rightarrow M_1$
18. $R_{(1)} \rightarrow M_3$
19. $M_1 \cup \overline{M_3} \rightarrow M_1$
20. $\overline{M_1} \cup M_2 \rightarrow M_3$ & Out $(= B_{(2)})$

- Calculate $D_{(2)}$ to $D_{(k-1)}$ and $B_{(3)}$ to $B_{(k)}$:

  Use the same instructions for calculating $D_{(1)}$ and $B_{(2)}$ except that $X_{(1)}$ and $R_{(1)}$ (and $D_{(1)}$ and $B_{(2)}$) are replaced by $X_{(i)}$ and $R_{(i)}$ (and $D_{(i)}$ and $B_{(i+1)}$) in each iteration, and in the beginning of an iteration the memory M_3 stores $B_{(i)}$ instead of $\overline{B_{(1)}}$, $i = 2, 3, .., k$.

Therefore, the total execution time in the DOCIP to complete this parallel subtraction is

$$t(k) \leq 20(k-1) + 3 = 20k - 17 = O(k).$$

4.3 Multiplication of Binary stack-coded Numbers

Let the result of the parallel multiplication be $X \times R = M = (M_{(2k-1)}, M_{(2k-2)}, ..., M_{(0)})$ (e.g. Fig. 12(e)). Since binary multiplication is equivalent to the addition of shifted versions of the multiplicand, applying the corresponding parallel operations, we can implement the parallel multiplication by the equations

$$P^{(0)} = (\underbrace{0, 0, ..., 0}_{k}, X_{(k-1)} \cap R_{(0)}, X_{(k-2)} \cap R_{(0)}, ..., X_{(0)} \cap R_{(0)}) \tag{44}$$

$$P^{(i)} = (\underbrace{0, 0, ..., 0}_{k-i}, X_{(k-1)} \cap R_{(i)}, X_{(k-2)} \cap R_{(i)}, ..., X_{(0)} \cap R_{(i)}, \underbrace{0, 0, ..., 0}_{i}) \tag{45}$$

$$X \times R = M = \sum_{i=0}^{k-1} P^{(i)} \equiv P^{(0)} + P^{(1)} + ... + P^{(k-1)} \tag{46}$$

where $i = 0, 1, ..., k-1$, and the addition $+$ is defined in subsection 4.1. Since this parallel multiplication requires at most $k-1$ additions, each addition takes $O(k)$ time for the DOCIP, and each $P^{(i)}$ can be generated in $O(k)$ time, the total execution time is $O(k^3)$.

5 Symbolic Substitution and Binary symbol-coded Arithmetic

Symbolic substitution was first considered as a means of utilizing the parallelism of optics by Huang [17]. Recently, the use of symbolic substitution as a basis for digital optical computing has been reported in [17]-[19] [24]-[31]. Special symbolic substitution rules can be applied to perform arithmetic operations and simulate a Turing machine [19]. Although symbolic substitution demonstrates the ability to solve any computable problem and performs many operations, we will formalize symbolic substitution by BIA algebraic symbols, demonstrate that symbolic substitution rules are particular BIA image transformations, and give the BIA formal notations of binary symbol-coded (symbolic substitution) arithmetic. We show that the symbolic substitution implementation of some operations is relatively complicated to other implementations.

5.1 BIA Representation of Symbolic Substitution

In this subsection we give the BIA equation for symbolic substitution and show how it can be implemented on the DOCIP machine. A symbolic substitution rule involves two steps: 1) recognizing the locations of a certain search-pattern within the 2-D binary input data, and 2) substituting a replacement-pattern wherever the search-pattern is recognized. We derive it by BIA in the following steps (illustrated in Fig. 13):

1. BIA Notations for Symbolic Substitution:
   - 2-D binary input data = image (bit plane) X
   - Symbol to be recognized (search-pattern) = reference image (or image pairs) R
   - Symbol to be replaced (replacement-pattern) = reference image Q
2. A Symbolic Substitution Rule:
   - Step 1. recognition of the search-pattern:
     (a) Foreground recognizer: the locations of a certain spatial search-pattern R_1 (defined by its foreground) within the *foreground* of the 2-D input data X can be recognized by the erosion operation of X and R_1:

$$X \ominus R_1 = \overline{\overline{X} \oplus \check{R}_1}. \tag{47}$$

     (b) Background recognizer: the locations of a certain spatial search-pattern R_2 within the *background* of the 2-D input data X can be recognized by the erosion of $\overline{X}$ and R_2:

$$\overline{X} \ominus R_2 = \overline{X \oplus \check{R}_2}. \tag{48}$$

(c) Full recognizer: by combining the two above steps, the locations of a certain spatial search-pattern $R = (R_1, R_2)$ (R_1 defines the foreground, and R_2 defines the background) within the 2-D input data X can be recognized by the hit or miss transform of X and R:

$$X \circledast R = (X \ominus R_1) \cap (\overline{X} \ominus R_2) = \overline{(\overline{X} \oplus \check{R}_1) \cup (X \oplus \check{R}_2)}. \tag{49}$$

- Step 2. substitution of the replacement-pattern:
  - Substituter: a new replacement-pattern Q can be substituted for R wherever the search-pattern R is recognized by the dilation of $X \circledast R$ by Q.
- Synthesis:
  - A complete symbolic sᵣbstitution rule is implemented by the hit or miss transform of X by R followed by the dilation by Q:

$$(X \circledast R) \oplus Q = ((X \ominus R_1) \cap (\overline{X} \ominus R_2)) \oplus Q = \overline{(\overline{X} \oplus \check{R}_1) \cup (X \oplus \check{R}_2)} \oplus Q. \tag{50}$$

- Optional masking:
  - An optional mask M can be used for controlling the block search region. A symbolic substitution rule can be modified as:

$$((X \circledast R) \cap M) \oplus Q. \tag{51}$$

By proper choice of M, the search can be made in overlapping, disjoint or non-contiguous blocks.

3. A symbolic substitution system (Fig. 14):

- To work with more than one rule (say p substitution rules) for practical applications, a symbolic substitution processor produces several copies of the input X, provides p different recognizer-substituter units, and then combines the outputs of various units to form a new output. Thus, a symbolic substitution system is implemented by

$$\bigcup_{i=1}^{p} (X \circledast R^{(i)}) \oplus Q^{(i)} \tag{52}$$

where $R^{(i)}$ and $Q^{(i)}$, $i = 1, 2, ..., p$, are the reference image pairs and replacement patterns in the i^{th} symbolic substitution rule. This, then, is the BIA formula for general symbolic substitution.

Hence, a general mathematical formalism of symbolic substitution has been developed. For a local search-pattern and replacement-pattern (i.e. $R_1, R_2, Q \subset N_{array}$ or $N_{hypercube}$), the DOCIP-array or DOCIP-hypercube can implement a symbolic substitution rule in four (or five with the optional mask) steps:

Assume start with X in M_1.

1. $\overline{M_1} \oplus \check{R}_1 \rightarrow M_2$
2. $M_1 \oplus \check{R}_2 \rightarrow M_3$
3. $M_2 \cup M_3 \rightarrow M_3$
4. $\overline{M_3} \oplus Q \rightarrow$ Out $(= (X \circledast R) \oplus Q)$

Let the pixels used in the substitution rule(s) of a symbolic substitution processor be the *neighborhood*, N_{ss}, of the processor. We see from the above steps that the DOCIP can simulate the symbolic substitution processor in constant time if the two machines have the same neighborhood. If N_{ss} is not a subset of the DOCIP neighborhood, then the simulation will take longer. In either case, it is not presently known how many steps it takes the symbolic substitution processor to simulate the DOCIP.

5.2 Binary symbol-coded (Symbolic Substitution) Arithmetic

A bit in a binary number is encoded in symbolically as pixels of an image (Fig. 15). In this subsection, we primarily concentrate on simple intensity coding: a logic value (0 or 1) is represented by a single pixel (dark or bright) (Fig. 15(a)), as in the binary row and stack-coded number representations, but the operands of binary numbers x_i and r_i are stored in the same input image X as shown in Fig. 16(a). The expected output images of symbolic substitution for binary addition and binary subtraction are shown in Fig. 16(b)-(c). To achieve these desired operations, the symbols associated with the operands are recognized and then replaced by new symbols associated with the results of the operation. Systems for implementing binary addition and subtraction are formalized and illustrated as examples of binary symbol-coded arithmetic below.

5.2.1 Addition of Binary symbol-coded Numbers

This parallel binary addition (Fig. 17) can be implemented with four symbolic substitution rules (Fig. 17(a)) [17] [18]. In the case of simple intensity coding, as we will show, Rule 1 is not necessary. The symbolic substitution system for simple intensity coding can be realized as

$$Y(t_0) = X \tag{53}$$

$$Y(t_{j+1}) = \bigcup_{i=1}^{4}((Y(t_j) \circledast R^{(i)}) \cap M) \oplus Q^{(i)} \tag{54}$$

where $Y(t_{k+1})$ is the result, $j = 0, 1, 2, ..., k+1$, k is word size (i.e. the number of bits in a operand); $R^{(i)} = (R_1^{(i)}, R_2^{(i)})$ and $Q^{(i)}$ are shown in Fig. 17(b) and represented as

1. $R_1^{(1)} = \phi$, $R_2^{(1)} = \bigcup_{i=0}^{1} B^i$, $Q^{(1)} = \phi$,
2. $R_1^{(2)} = I$, $R_2^{(2)} = B$, $Q^{(2)} = I$,
3. $R_1^{(3)} = B$, $R_2^{(3)} = I$, $Q^{(3)} = I$,
4. $R_1^{(4)} = \bigcup_{i=0}^{1} B^i$, $R_2^{(4)} = \phi$, $Q^{(4)} = A^{-1}B$.

Here the null image ϕ and the elementary images are as defined in subsection 2.1; the mask M (Fig. 17(c)), used for controlling the block search region, is the image corresponding to the coordinates of the origins (lower-lefter pixels) of the input symbols in the input image X. An example is given in Fig. 17(d). Note that $Q^{(1)} = \phi$ implies

$$((Y(t_j) \circledast R^{(1)}) \cap M) \oplus Q^{(1)} = \phi, \tag{55}$$

so that

$$\begin{aligned} Y(t_{j+1}) &= \bigcup_{i=1}^{4}((Y(t_j) \circledast R^{(i)}) \cap M) \oplus Q^{(i)} \\ &= \bigcup_{i=2}^{4}((Y(t_j) \circledast R^{(i)}) \cap M) \oplus Q^{(i)} \\ &= \bigcup_{i=2}^{4}(\overline{(\overline{Y}(t_j) \oplus \check{R}_1^{(i)}) \cup (Y(t_j) \oplus \check{R}_2^{(i)})} \cap M) \oplus Q^{(i)}. \end{aligned} \tag{56}$$

Thus, for simple intensity coding of symbolic substitution, we can reduce the four rules of binary addition to only three rules. However, this reduction of complexity cannot be applied to dual-rail or six-pixel coding.

When implemented on the DOCIP, this addition requires at most $k+1$ iterations, each iteration requiring two union operations of three results of symbolic substitution rules, and each rule is realized within five steps as shown in subsection 5.1. Thus, the total execution time in the DOCIP is

$$t(k) \leq (3 \times 5 + 2)(k+1) = 17(k+1) = O(k).$$

When using 2 or 6 pixels to represent a logic value (Fig. 15(b)-(c)), we can formalize symbolic substitution addition as

- Dual-rail coding (Fig. 15(b)) [19][20]: we can implement a full recognition with only a background recognizer (or foreground recognizer)

  $$\begin{aligned} Y(t_{j+1}) &= \bigcup_{i=1}^{4}((Y(t_j) \circledast R^{(i)}) \cap M) \oplus Q^{(i)} \\ &= \bigcup_{i=1}^{4}(\overline{(\overline{Y}(t_j) \oplus \check{R}_1{}^{(i)}) \cup (Y(t_j) \oplus \check{R}_2{}^{(i)})} \cap M) \oplus Q^{(i)} \\ &= \bigcup_{i=1}^{4}(\overline{Y(t_j) \oplus \check{R}_2{}^{(i)}} \cap M) \oplus Q^{(i)} \end{aligned} \tag{57}$$

  where $j = 0, 1, 2, ..., k$; $R^{(i)} = (R_1^{(i)}, R_2^{(i)})$ and $Q^{(i)}$ are shown in Fig. 18(a) and represented by elementary images as

  1. $R_1^{(1)} = I \cup B^2$, $R_2^{(1)} = B \cup B^3$, $Q^{(1)} = I \cup A^{-1}B^2$,
  2. $R_1^{(2)} = \bigcup_{i=1}^{2} B^i$, $R_2^{(2)} = I \cup B^3$, $Q^{(2)} = B \cup A^{-1}B^2$,
  3. $R_1^{(3)} = I \cup B^3$, $R_2^{(3)} = \bigcup_{i=1}^{2} B^i$, $Q^{(3)} = B \cup A^{-1}B^2$,
  4. $R_1^{(4)} = B \cup B^3$, $R_2^{(4)} = I \cup B^2$, $Q^{(4)} = I \cup A^{-1}B^3$;

  and the mask M is shown in Fig. 18(b). Since

  $$(\overline{Y}(t_j) \oplus \check{R}_1{}^{(i)}) \cup (Y(t_j) \oplus \check{R}_2{}^{(i)}) = (\overline{Y}(t_j) \oplus \check{R}_1{}^{(i)}) = (Y(t_j) \oplus \check{R}_2{}^{(i)}), \tag{58}$$

  for the dual-rail coding, $R^{(i)}$ can be represented by only its foreground $R_1^{(i)}$ or background $R_2^{(i)}$. For implementation on the DOCIP, this algorithm requires four rules, and each rule involves two dilations and one union or intersection. Because they may be not included in N_{array} or $N_{hypercube}$, each dilation of $R_2^{(i)}$ or $Q^{(i)}$ is implemented by 2-4 steps for the DOCIP-array8 and 1-2 steps for the DOCIP-hypercube8. The total execution time is bounded by $28(k+1)$ for the DOCIP-array8 and $18(k+1)$ for the DOCIP-hypercube8. Moreover, it requires more difficult dual-rail coding and doubles the device area.

- Six-pixel coding (Fig. 15(c)) [30]: the mask M is not needed and

$$Y(t_{j+1}) = \bigcup_{i=1}^{4} (Y(t_j) \circledast R^{(i)}) \oplus Q^{(i)} \tag{59}$$

where $j = 0, 1, 2, ..., k$, k is the word size; $R^{(i)} = (R_1^{(i)}, R_2^{(i)})$ and $Q^{(i)}$ are shown in Fig. 21 and are represented as

1. $R_1^{(1)} = I \cup AB \cup B^2 \cup AB^3$, $R_2^{(1)} = B \cup \cup B^3 \cup A \cup AB^2 \cup (\bigcup_{i=0}^{3} A^2B^i)$, $Q^{(1)} = A^{-3}B^2 \cup A^{-2}B^3 \cup I \cup AB$,
2. $R_1^{(2)} = (\bigcup_{i=1}^{2} B^i) \cup A \cup A^3$, $R_2^{(2)} = I \cup B^3 \cup (\bigcup_{i=1}^{2} AB^i) \cup (\bigcup_{i=0}^{3} A^2B^i)$, $Q^{(2)} = A^{-3}B^2 \cup A^{-2}B^3 \cup B \cup A$,
3. $R_1^{(3)} = I \cup B^3 \cup (\bigcup_{i=1}^{2} AB^i)$, $R_2^{(3)} = (\bigcup_{i=1}^{2} B^i) \cup A \cup A^3 \cup (\bigcup_{i=0}^{3} A^2B^i)$, $Q^{(3)} = A^{-3}B^2 \cup A^{-2}B^3 \cup B \cup A$,
4. $R_1^{(4)} = B \cup B^3 \cup A \cup AB^2$, $R_2^{(4)} = I \cup B^2 \cup AB \cup AB^3 \cup (\bigcup_{i=0}^{3} A^2B^i)$, $Q^{(4)} = A^{-3}B^3 \cup A^{-2}B^2 \cup I \cup AB$.

The six-pixel coding removes the need for the mask M, but requires more difficult encoding, more difficult implementation of the hit or miss transform by $R^{(i)}$ and dilation by $Q^{(i)}$, and six times the hardware area. Addition on the DOCIP-array or DOCIP-hypercube using six-pixel coding takes much more time (on the order of ten times) than simple intensity coding or dual-rail coding.

5.2.2 Subtraction of Binary symbol-coded Numbers

Similar to addition, we generally use 4 symbolic substitution rules (Fig. 20(a)), but Rule 1 and Rule 4 are not necessary for simple intensity coding. The symbolic substitution system using simple intensity coding for binary subtraction can be realized as

$$Y(t_0) = X \tag{60}$$

$$\begin{aligned} Y(t_{j+1}) &= \bigcup_{i=1}^{4}((Y(t_j) \circledast R^{(i)}) \cap M) \oplus Q^{(i)} \\ &= \bigcup_{i=1}^{4}(\overline{(\overline{Y}(t_j) \oplus \check{R}_1{}^{(i)}) \cup (Y(t_j) \oplus \check{R}_2{}^{(i)})} \cap M) \oplus Q^{(i)} \\ &= \bigcup_{i=2}^{3}(\overline{(\overline{Y}(t_j) \oplus \check{R}_1{}^{(i)}) \cup (Y(t_j) \oplus \check{R}_2{}^{(i)})} \cap M) \oplus Q^{(i)} \end{aligned} \tag{61}$$

where $Y(t_{k+1})$ is the result of the subtraction, $j = 0, 1, 2, ..., k$, k is word size (i.e. the number of bits in a operand); $R^{(i)} = (R_1^{(i)}, R_2^{(i)})$ and $Q^{(i)}$ are shown in Fig. 20(b) and represented as

1. $R_1^{(1)} = \phi$, $R_2^{(1)} = \bigcup_{i=0}^{1} B^{-i}$, $Q^{(1)} = \phi$,
2. $R_1^{(2)} = B^{-1}$, $R_2^{(2)} = I$, $Q^{(2)} = I \cup A^{-1}B^{-1}$,
3. $R_1^{(3)} = I$, $R_2^{(3)} = B^{-1}$, $Q^{(3)} = I$,
4. $R_1^{(4)} = \bigcup_{i=0}^{1} B^{-i}$, $R_2^{(4)} = \phi$, $Q^{(4)} = \phi$

where the null image ϕ and the elementary images are as defined in subsection 2.1; and the mask M (Fig. 20(c)) is a shifting of the mask for binary addition. Because $Q^{(1)}$ and $Q^{(4)}$ are null images, and the dilation of a null image is a null image, Rule 1 and Rule 4 are not needed for simple intensity coding. Fig. 20(d) gives an example. The execution time for the DOCIP is

$$t(k) \leq 11(k+1) = O(k).$$

Similar to binary addition, we can develop symbolic substitution binary subtraction algorithms with BIA representations for coding a symbol with two or six pixels. However, four symbolic substitution rules are still required because $Q^{(1)}$ and $Q^{(4)}$ will not be equal to the null image. The DOCIPs take approximately the same execution time for binary subtraction using dual-rail or six-pixel coding as for binary addition.

6 Complexity of Parallel Optical Binary Arithmetic

We have shown that BIA offers a general tool for mapping serial binary arithmetic into different forms of parallel binary arithmetic (including binary row(or column)-coding, binary stack-coding, and three coding techniques for symbolic substitution arithmetic) in a precise and compact way. The complexity of parallel addition and subtraction of two $N \times N$ arrays of binary numbers (each number with k-bit length) for these different number representations are compared in Table 1 and Table 2. Binary row(or column)-coded arithmetic requires the smallest number O of fundamental operations. Binary stack-coded arithmetic requires the lowest number of processing elements (or cells) P and the smallest overall $O \times P$ complexity (assume each parallel fundamental operation corresponds to P processing elements executing in parallel). For the normal case in which the word size is larger than one and much smaller than the image size ($1 < k \ll N$), binary row(or column)-coded arithmetic can be implemented in the DOCIP with the fastest computation speed (assume the DOCIP can input all operands in an image at a time). The complexity of binary symbol-coded (symbolic substitution) arithmetic in general is in all cases higher than that of binary row(or column)-coded and binary stack-coded arithmetic. For implementing symbolic substitution algorithms on the DOCIPs, the simple intensity coding is superior to the other symbol coding techniques.

7 Conclusion

Binary image algebra (BIA) is demonstrated to be a general technique for developing and formulating parallel numerical and non-numerical computation algorithms for digital optical computers. The DOCIP is a simple optical architecture for effectively implementing BIA. Symbolic substitution is a subset of BIA and can be formalized in compact BIA expressions. Three different techniques for parallel optical binary arithmetic, based on binary row(or column)-coding, binary stack-coding, and binary symbol-coding (symbolic substitution), are illustrated for implementation on the DOCIP. Binary row-coding arithmetic has fast DOCIP execution and binary stack-coding arithmetic requires the lowest number of computations $O \times P$. In summary, BIA and the DOCIP represent a simple yet powerful parallel digital optical algorithmic and architectural technique for both numerical and non-numerical applications.

Acknowledgements

The authors would like to thank P. Chavel and E. K. Blum for their contributions to this research.

References

[1] A. A. Sawchuk and T. C. Strand, "Digital Optical Computing," *Proc. IEEE*, Vol.72, pp. 758-779,1984.

[2] *Proc. IEEE*, Special Issue on Optical Computing, Vol. 72, No. 7, 1984.

[3] A. Huang, et al, "Optical Computation Using Residue Arithmetic," *Applied Optics*, 18, p. 149, 1979.

[4] D. Psaltis and D. Casasent, " Optical Residue Arithmetic: A Correlation Approach," *Applied Optics*, 18, p. 163, 1979.

[5] F. A. Horrigan and W. W. Stoner, "Reside-Based Optical Prosessor," *Proc. SPIE*, 185, 19, 1979.

[6] A. Tai, et al, "Optical Residue Arithmetic Computer with Programmable Computation Modules," *Applied Optics*, 18, p. 2812 , 1979.

[7] G. Abraham, "Multiple-valued Logic for Optoelectronics," *Optical Engineering*, Vol. 25, No. 1, p.3, 1986.

[8] T. T. Tao and D. M. Campell, "Multiple-valued Logic: An implementation," *Optical Engineering*, Vol. 25, No. 1, p.14, 1986.

[9] R. Arrathoon and S. Kozaitis, " Shadow Casting for Multiple-valued Associative Logic," *Optical Engineering*, Vol. 25, No. 1, p. 29, 1986.

[10] S. L. Hurst, "Multiple-valued Threshold Logic: Its Status and Its Realization," *Optical Engineering*, Vol. 25, No. 1, p. 44, 1986.

[11] B. K. Jenkins, et al, "Sequential Optical Logic Implementation," *Applied Optics*, Vol. 23, No. 19, pp. 3455-3464, 1984.

[12] B. K. Jenkins, et al, "Architectural Implications of A Digital Optical Processor," *Applied Optics*, Vol. 23, No. 19, pp. 3465-3474, 1984.

[13] K. S. Huang, B. K. Jenkins, A. A. Sawchuk, "Binary Image Algebra and Digital Optical Cellular Image Processors", *Topical Meeting on Optical Computing*, Technical Digest Series 1987, Vol. 11, Optical Society of America, Washington, D.C., pp. 20-23, 1987.

[14] K. S. Huang, B. K. Jenkins, A. A. Sawchuk, "A Cellular Hypercube Architecture for Image Processing", *Proc. Soc. Photo-Opt. Instr. Eng.*, Vol. 829, August, 1987.

[15] K. S. Huang, B. K. Jenkins, A. A. Sawchuk, "Optical Cellular Logic Architectures Based on Binary Image Algebra ", *Proc. IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Machine Intelligence*, Seattle, October, pp. 19-26, 1987.

[16] K. S. Huang, B. K. Jenkins, A. A. Sawchuk, "Binary Image Algebra and Optical Cellular Logic Processor Design," submitted to *Computer Vision, Graphics, and Image Processing.*

[17] A. Huang, "Parallel Algorithms for Optical Digital Computers," in *Technical Digest, IEEE Tenth International Optical Computing Conference*, pp. 13-17, 1983.

[18] K. Brenner and A. Huang, "An Optical Processor Based on Symbolic Substitution", *Conference Proceeding of the Topical Meeting on Optical Computing*, Optical Society of America, March 18, pp. WA4.1-WA4.3, 1985.

[19] K.-H. Brenner, A. Huang, and N. Streibl, "Digital Optical Computing with Symbolic Substitution," *Applied Optics*, Vol. 25, pp. 3054-3060, 1986.

[20] K. S. Huang, B. K. Jenkins, A. A. Sawchuk, "Binary Image Algebra Representations of Optical Cellular Logic and Symbolic Substitution", *1987 Annual Meeting*, Optical Society of America, Rochester, New York, Oct., 1987.

[21] D. Psaltis and R. A. Athale, "High Accuracy Computation with Linear Analog Optical Systems: A Critical Study", *Applied Optics*, Vol. 25, No. 18, pp. 3071-3077, 1986.

[22] K. S. Huang, B. K. Jenkins, A. A. Sawchuk, "Programming A Digital Optical Cellular Image Processor", *1987 Annual Meeting*, Optical Society of America, Rochester, New York, Oct., 1987.

[23] B. K. Jenkins and A. A. Sawchuk, "Optical Cellular Logic Architectures for Image Processing ", *Proc. IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, Florida, Nov., pp. 61-65, 1985.

[24] K.-H. Brenner, "New Implementation of Symbolic Substitution," *Applied Optics*, Vol. 25, pp. 3061-3064, 1986.

[25] K.-H. Brenner and G. Stucke "Programmable Optical Processor Based on Symbolic Substitution," *Topical Meeting on Optical Computing*, Technical Digest Series 1987, Vol. 11, Optical Society of America, Washington, D.C., pp. 6-8, 1987.

[26] J. N. Mait and K.-H. Brenner, "Optical Systems for Symbolic Substitution," *Topical Meeting on Optical Computing*, Technical Digest Series 1987, Vol. 11, Optical Society of America, Washington, D.C., pp. 12-15, 1987.

[27] T. J. Cloonan, "Strengths and Weaknesses of Optical Architectures Based on Symbolic Substitution," *Topical Meeting on Optical Computing*, Technical Digest Series 1987, Vol. 11, Optical Society of America, Washington, D.C., pp. 12-15, 1987.

[28] C. D. Capps, R. A. Falk and T. L. Houk "Optical Arithmetic/Logic Unit Based on Residue Number Theory and Symbolic Substitution", *Topical Meeting on Optical Computing*, Technical Digest Series 1987, Vol. 11, Optical Society of America, Washington, D.C., pp. 62-65, 1987.

[29] P. A. Ramamoorthy and S. Antony "Optical MSD Adder Using Polarization Coded Symbolic Substitution", *Topical Meeting on Optical Computing*, Technical Digest Series 1987, Vol. 11, Optical Society of America, Washington, D.C., pp. 111-114, 1987.

[30] Ho-In Jeon, "Digital Optical Processor Based on Symbolic Substitution Using Matched Filtering", *Topical Meeting on Optical Computing*, Technical Digest Series 1987, Vol. 11, Optical Society of America, Washington, D.C., pp. 115-118, 1987.

[31] M. T. Taso, et al, "Symbolic Substitution Using ZnS Interference Filters", *Optical Engineering*, Vo. 26, No. 1, January, pp. 41-44, 1987.

$$\begin{array}{|ccccccc|}\hline 0&0&0&0&0&0&0\\ 0&0&1&1&1&1&0\\ 0&0&0&1&1&1&0\\ 0&0&0&0&1&1&0\\ 0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ \hline\end{array}$$

Tested Image X

$$\begin{array}{|ccccccc|}\hline 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&1&1&1&0&0\\ 0&0&1&1&1&0&0\\ 0&0&1&1&1&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ \hline\end{array}$$

Reference Image R

$$\begin{array}{|ccccccc|}\hline 1&1&1&1&1&1&1\\ 1&1&0&0&0&0&1\\ 1&1&1&0&0&0&1\\ 1&1&1&1&0&0&1\\ 1&1&1&1&1&0&1\\ 1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1\\ \hline\end{array}$$

Complement $\overline{X}$

$$\begin{array}{|ccccccc|}\hline 0&0&0&0&0&0&0\\ 0&0&1&1&1&1&0\\ 0&0&1&1&1&1&0\\ 0&0&1&1&1&1&0\\ 0&0&1&1&1&1&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ \hline\end{array}$$

Union $X \cup R$

$$\begin{array}{|ccccccc|}\hline 0&1&1&1&1&1&1\\ 0&1&1&1&1&1&1\\ 0&1&1&1&1&1&1\\ 0&0&1&1&1&1&1\\ 0&0&0&1&1&1&1\\ 0&0&0&0&1&1&1\\ 0&0&0&0&0&0&0\\ \hline\end{array}$$

Dilation $X \oplus R$

Figure 1: An example of fundamental operations: complement $^{-}$, union $\cup$, and dilation $\oplus$.

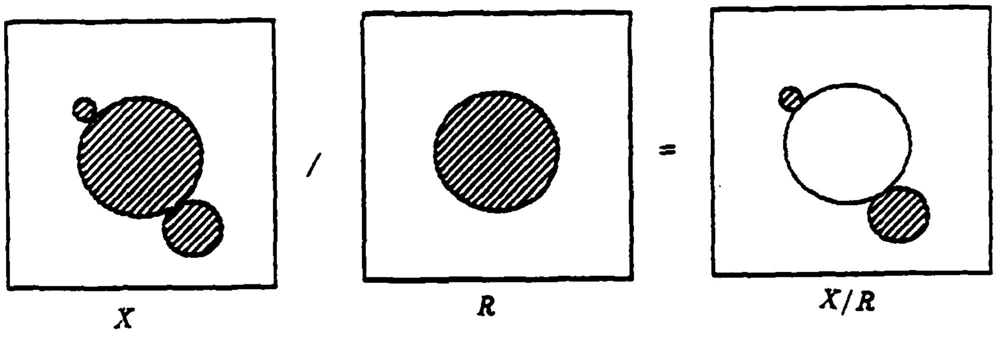


Figure 2(a): Difference.

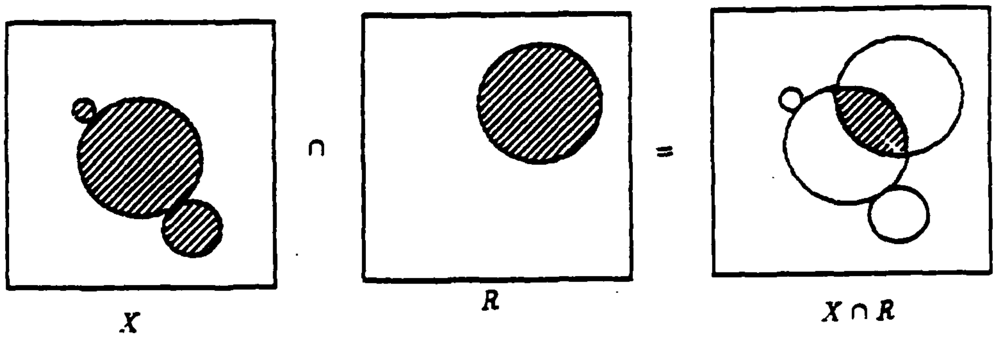


Figure 2(b): Intersection.

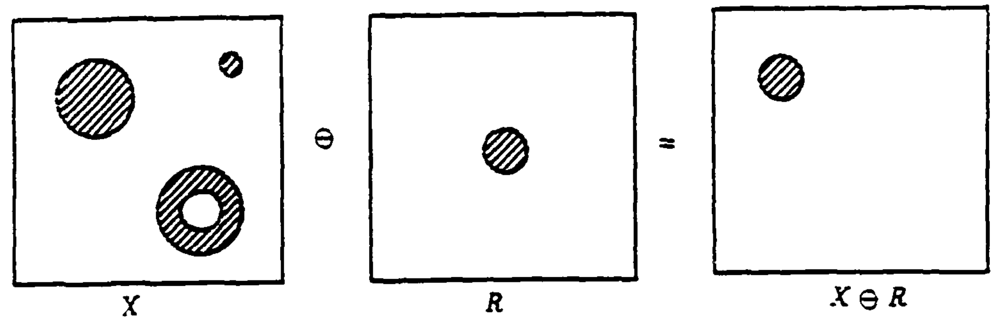


Figure 2(c): Erosion.

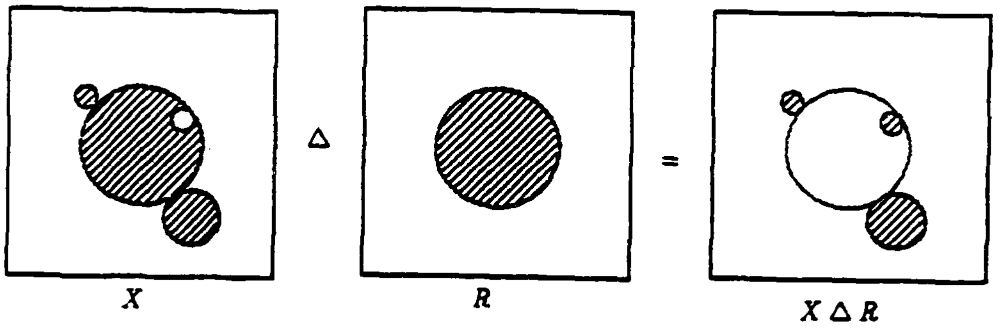


Figure 2(d): Symmetric difference.

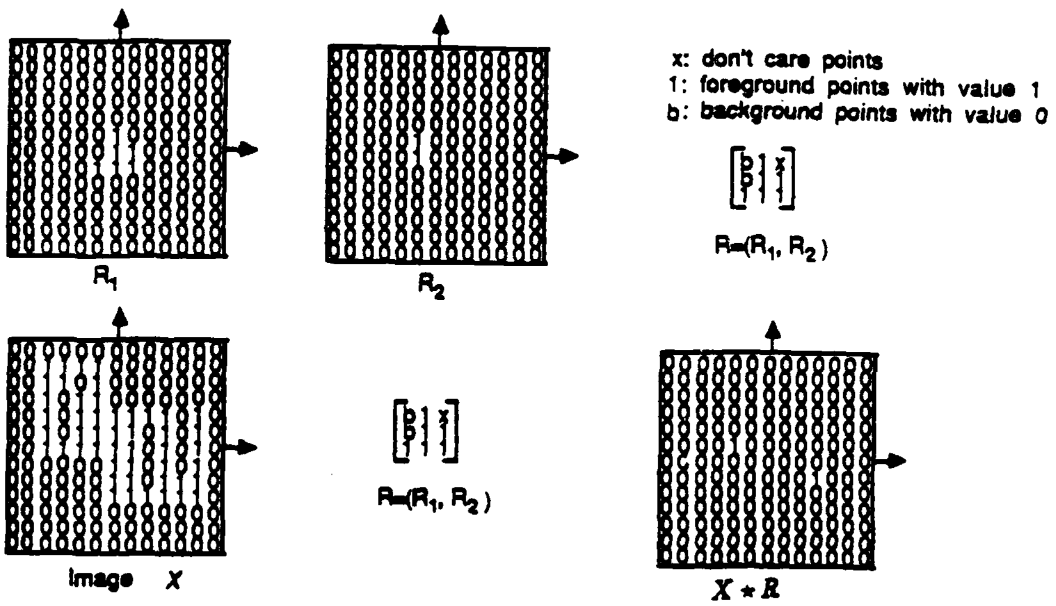

x: don't care points
1: foreground points with value 1
b: background points with value 0
R=(R₁, R₂)
R₁
R₂
Image X
R=(R₁, R₂)
X * R

Figure 2(e): Hit or miss transform (template matching).

Figure 2: Some standard derived image operations. The shaded regions in (a)-(d) correspond to pixels with value 1.

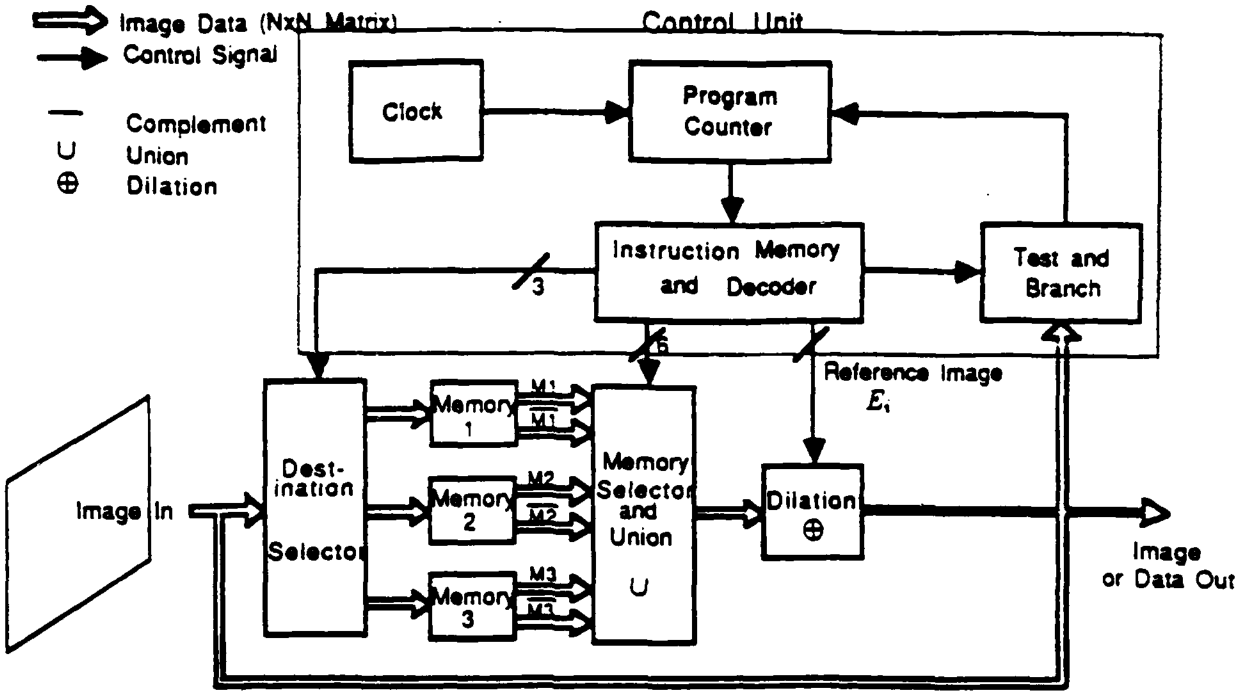


Figure 3: A digital optical cellular image processor (DOCIP) architecture — one implementation of binary image algebra (BIA). The DOCIP-array requires 9 (or 5) control bits for reference image E_i. The DOCIP-hypercube requires $O(logN)$ control bits for reference image E_i.

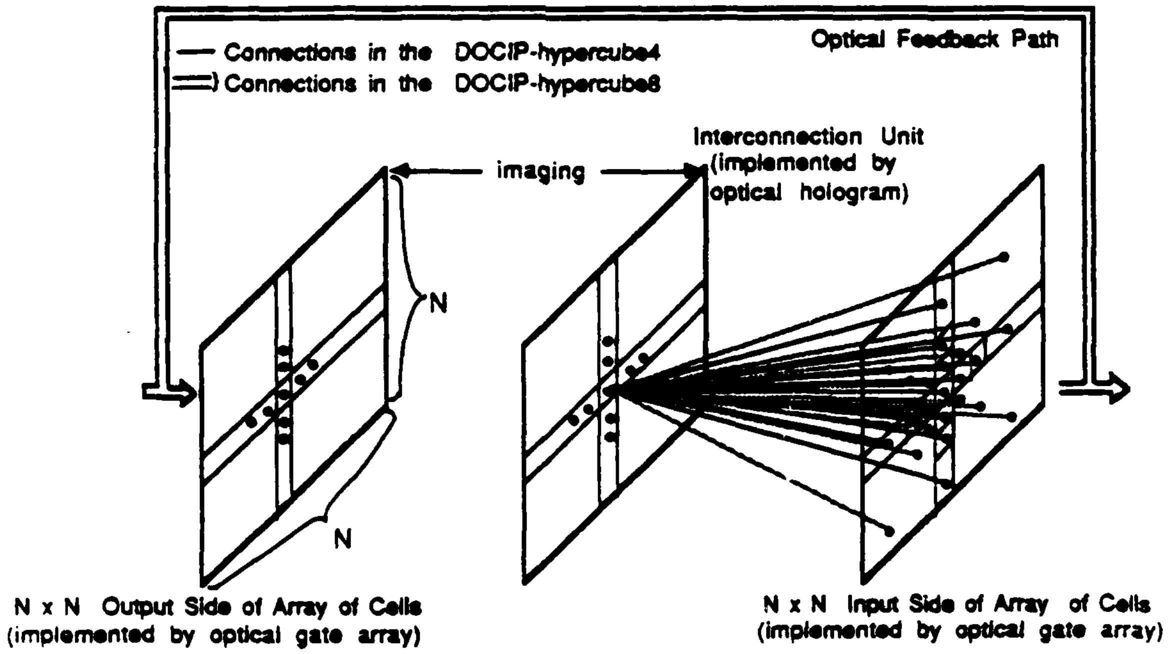

Connections in the DOCIP-hypercube4
Connections in the DOCIP-hypercube8
Optical Feedback Path
Interconnection Unit (implemented by optical hologram)
imaging
N
N
N x N Output Side of Array of Cells (implemented by optical gate array)
N x N Input Side of Array of Cells (implemented by optical gate array)

Figure 4: An optical 4-directed or 8-directed cellular hypercube (DOCIP-hypercube4 or DOCIP-hypercube8). Each cell connects with cells in the 4 directions or 8 directions at distances $1, 2, 4, 8, ..., 2^k$ from itself by optical 3-D interconnections.

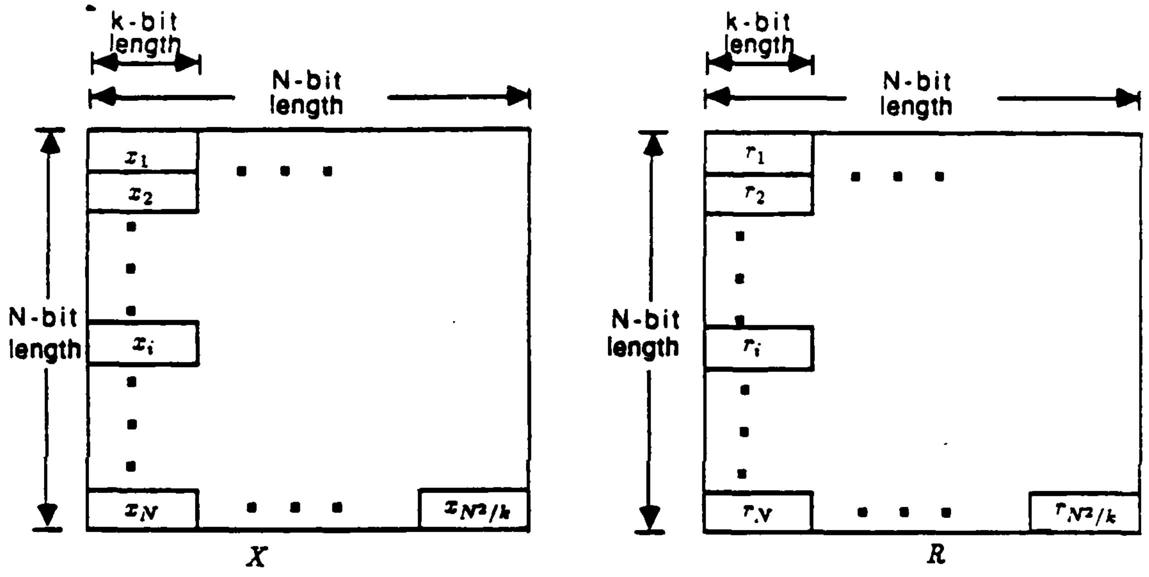
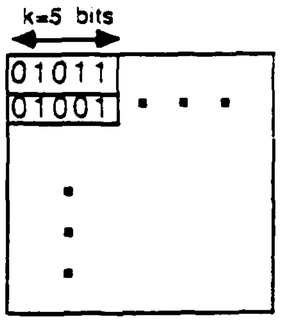
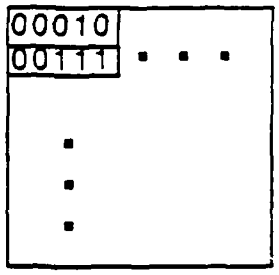


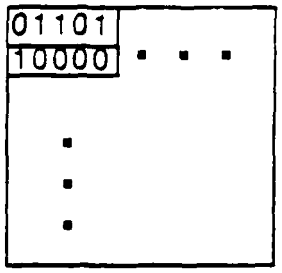
Figure 5: Binary row-coded numbers.



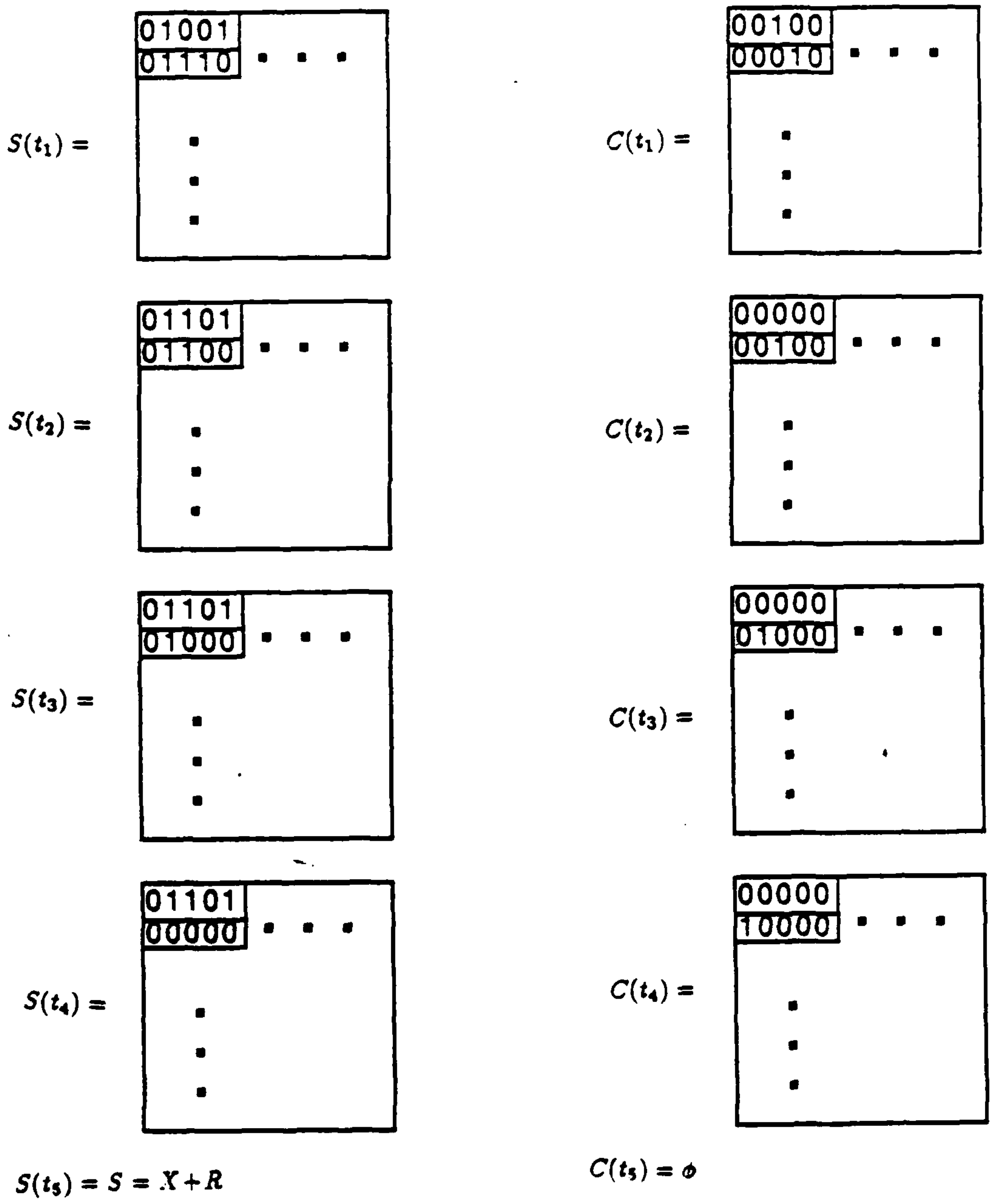
(a): An image X of operands.



(b): An image R of other operands.

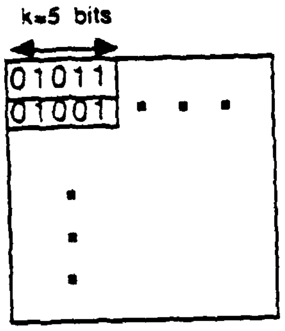


(c): The output $X + R$.

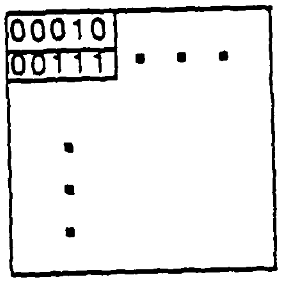


(d): The procedure for parallel addition $X + R$.

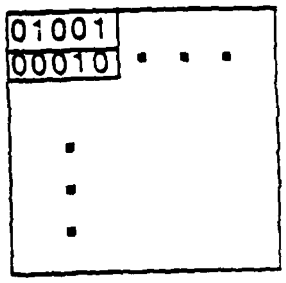
Figure 6: Parallel addition of binary row-coded numbers.



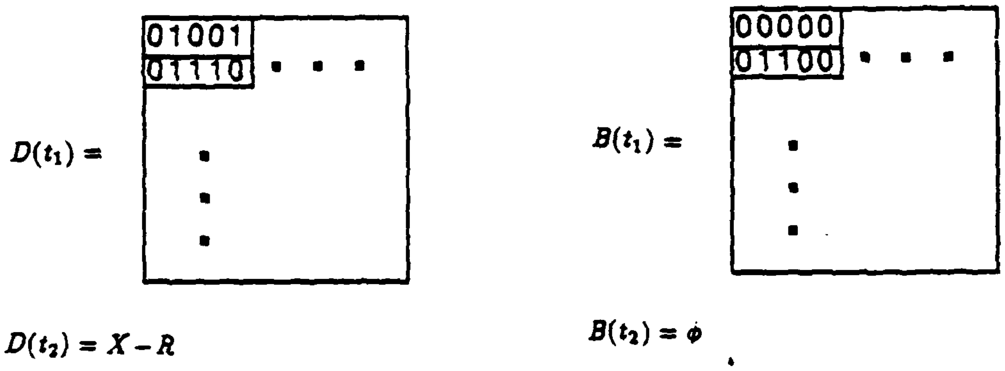
(a): An image X of operands.



(b): An image R of other operands.

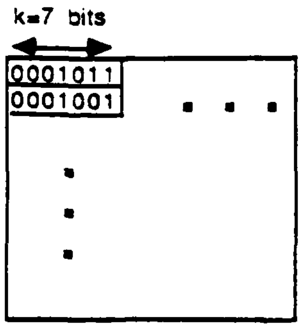


(c): The output $X - R$.

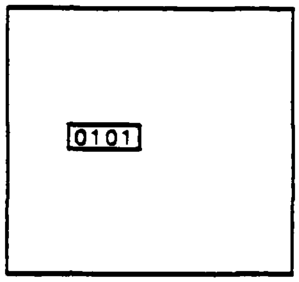


(d): The procedure for parallel subtraction $X - R$.

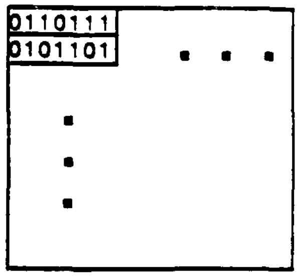
Figure 7: Parallel subtraction of binary row-coded numbers.



(a): An image X of operands.

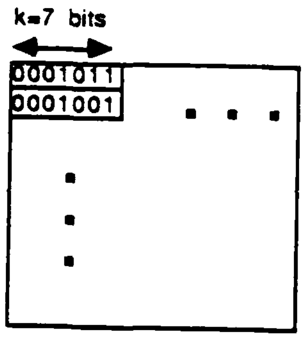


(b): An image R_r containing only a single number.

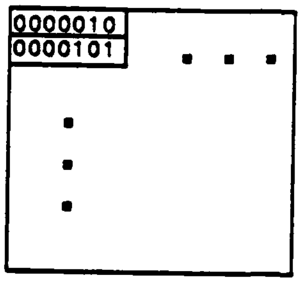


(c): The output $X \cdot R_r$

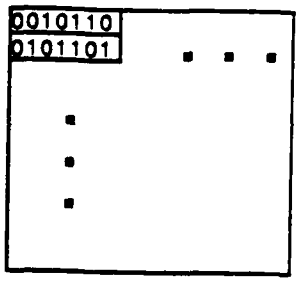
Figure 8: Parallel (matrix-constant) multiplication of binary row-coded numbers.



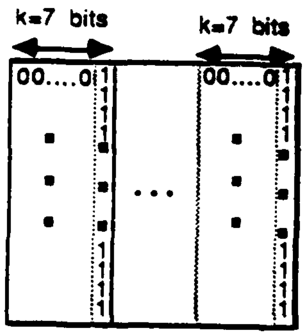
(a): An image X of operands.



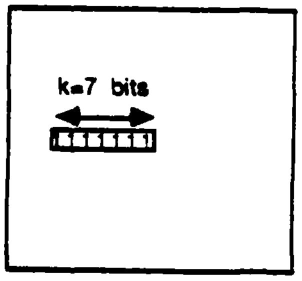
(b): An image R of other operands.



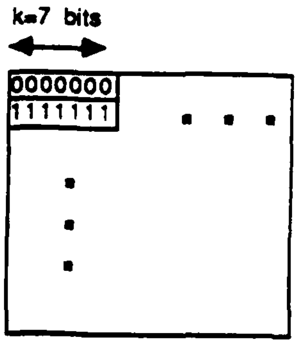
(c): The output $X \times R$.



(d): The mask M.



(e): The image $\bigcup_{j=0}^{k-1} A^{-j}$.



(f): The image $(R \cap M) \oplus \bigcup_{j=0}^{k-1} A^{-j}$.

Figure 9: Parallel (element-element) multiplication of binary row-coded numbers.

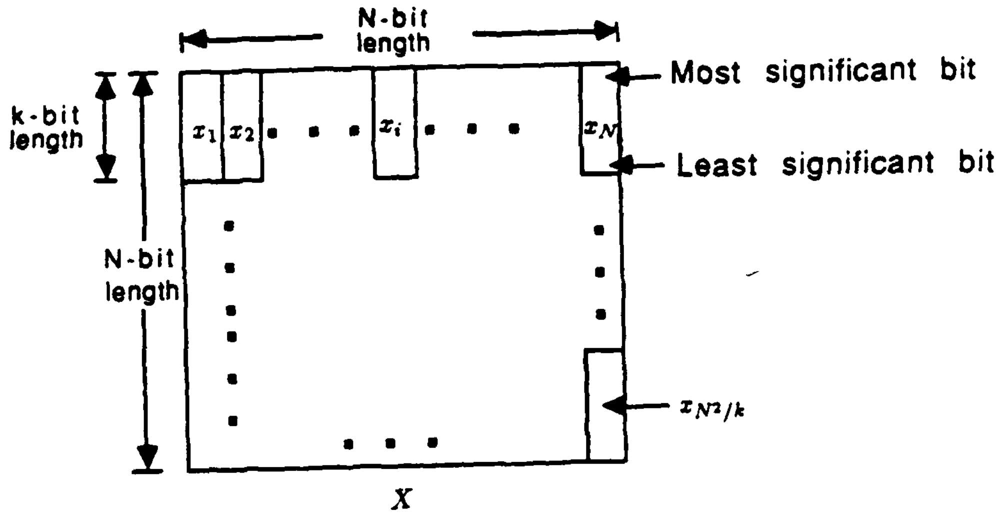


Figure 10: Binary column-coded numbers.

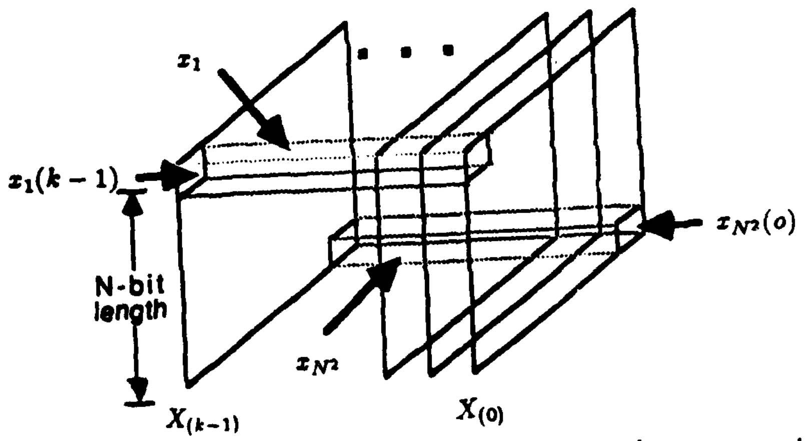


Figure 11: Binary stack-coded numbers. $x_i(m)$ represents the m^{th} bit of the i^{th} number in the image plane. $X_{(0)}$ represents the image plane of least significant bits and $X_{(k-1)}$ represents the image plane of most significant bits.

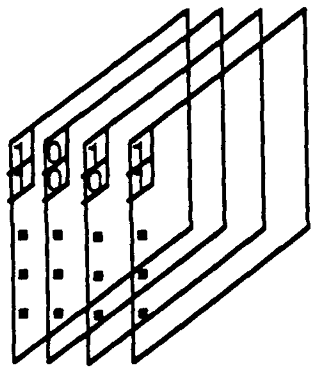


Figure 12(a): A sequence of images $X = (X_{(3)}, X_{(2)}, X_{(1)}, X_{(0)})$.

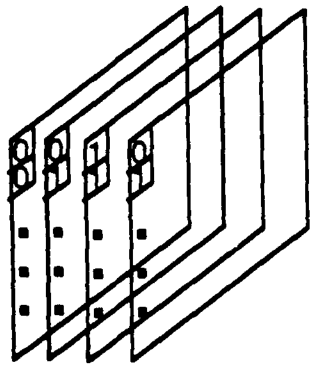


Figure 12(b): A sequence of images $R = (R_{(3)}, R_{(2)}, R_{(1)}, R_{(0)})$.

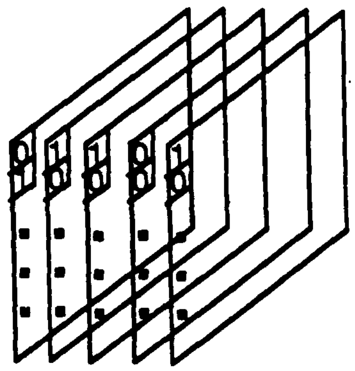


Figure 12(c): The sum $X+R = (S_{(4)}, S_{(3)}, S_{(2)}, S_{(1)}, S_{(0)})$.

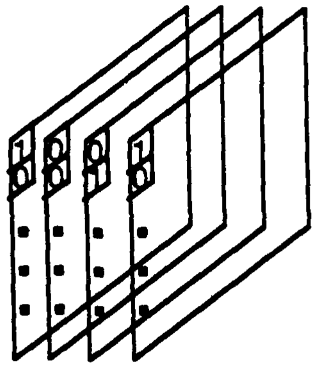


Figure 12(d): The difference $D = X - R = (D_{(3)}, D_{(2)}, D_{(1)}, D_{(0)})$.

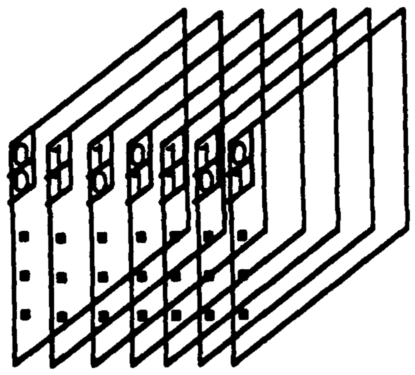


Figure 12(e): The product $M = X \times R = (M_{(7)}, M_{(6)}, ..., M_{(0)})$.

Figure 12: Parallel arithmetic with binary stack-coded numbers.

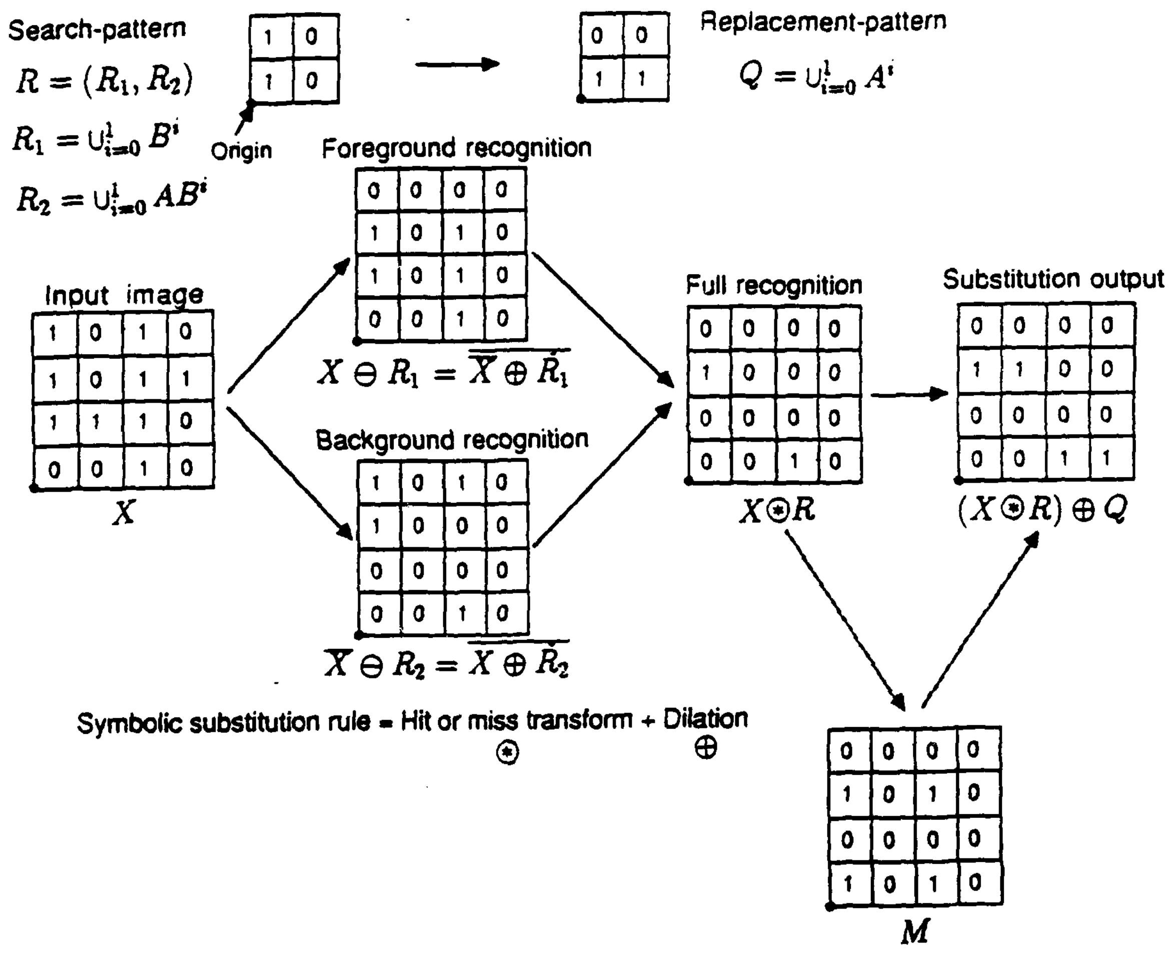

Search-pattern
$R = (R_1, R_2)$
$R_1 = \cup_{i=0}^{1} B^i$
$R_2 = \cup_{i=0}^{1} AB^i$
Origin
Replacement-pattern
$Q = \cup_{i=0}^{1} A^i$
Foreground recognition
$X \ominus R_1 = \overline{\overline{X} \oplus \check{R}_1}$
Background recognition
$\overline{X} \ominus R_2 = \overline{X \oplus \check{R}_2}$
Input image
X
Full recognition
$X \circledast R$
Substitution output
$(X \circledast R) \oplus Q$
M
Symbolic substitution rule = Hit or miss transform + Dilation
$\circledast$
$\oplus$

Figure 13: BIA representation of symbolic substitution. The optional mask M is tor controlling the block seach region.

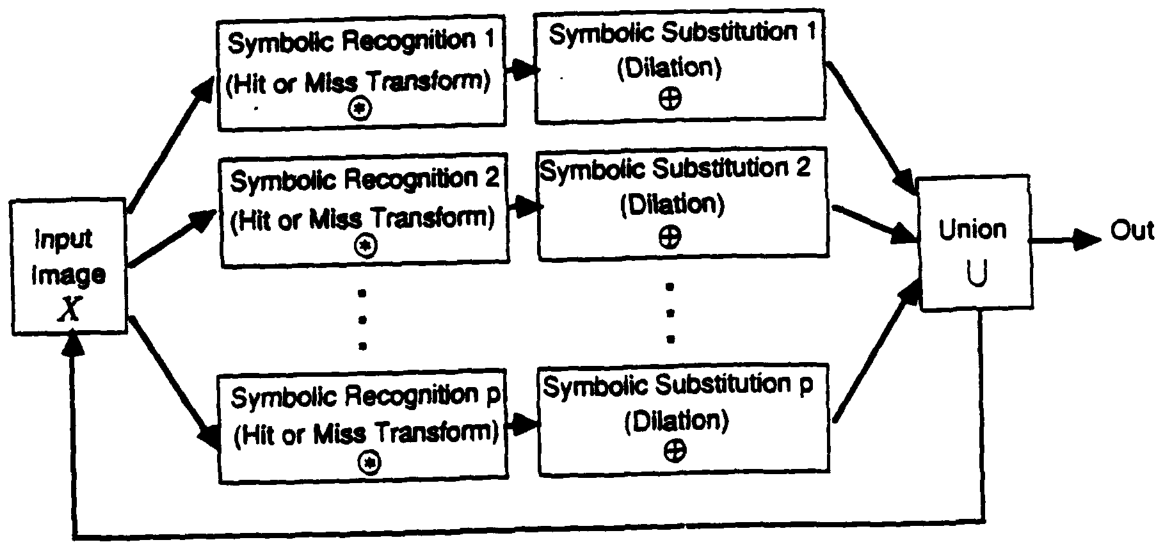

Input Image X
Symbolic Recognition 1 (Hit or Miss Transform) $\circledast$
Symbolic Recognition 2 (Hit or Miss Transform) $\circledast$
Symbolic Recognition p (Hit or Miss Transform) $\circledast$
Symbolic Substitution 1 (Dilation) $\oplus$
Symbolic Substitution 2 (Dilation) $\oplus$
Symbolic Substitution p (Dilation) $\oplus$
Union $\cup$
Out

Figure 14: A symbolic substitution system with p symbolic substitution rules.

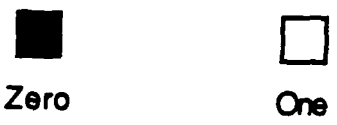


Figure 15(a): The simple intensity coding of zero and one (a bit is a pixel).

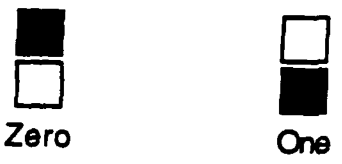


Figure 15(b): The dual-rail coding of zero and one (a bit is encoded as two pixels) (adapted from [18][19]).

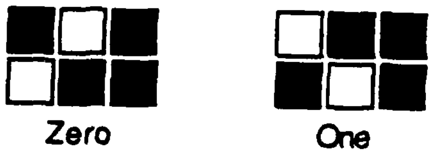


Figure 15(c): The six-pixel coding of zero and one (a bit with value zero or one is encoded as six pixels) (adapted from [30]).

Figure 15: A bit encoded as a symbol.

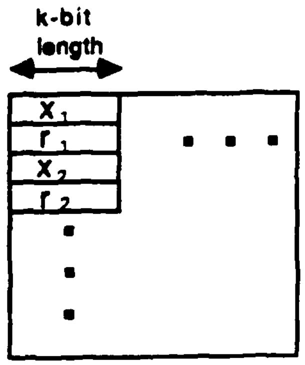
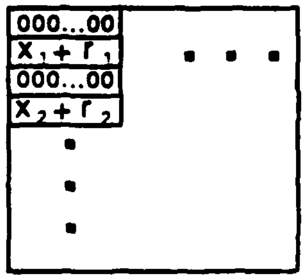


Figure 16(a): The input image X contains the operands x_i and the other operands r_i.



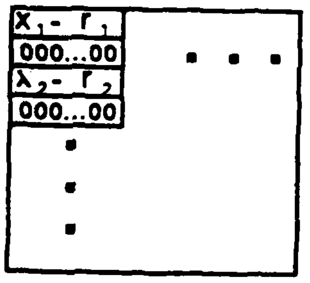


Figure 16(b): The output of parallel addition.

Figure 16(c): The output of parallel subtraction.

Figure 16: Binary symbol-coding (symbolic substitution) binary arithmetic).

```
                10110
               +10011
Carry bits      10010
Sum bits        00101
Carry bits      0010
Sum bits        100001
Carry bits      000
Sum bits        101001
```

Rule 1. $\substack{0\\0} \longrightarrow {}^{0}{}_{0}$

Rule 2. $\substack{0\\1} \longrightarrow {}^{0}{}_{1}$

Rule 3. $\substack{1\\0} \longrightarrow {}^{0}{}_{1}$

Rule 4. $\substack{1\\1} \longrightarrow {}^{1}{}_{0}$

Figure 17(a): Four symbolic substitution rules for addition.

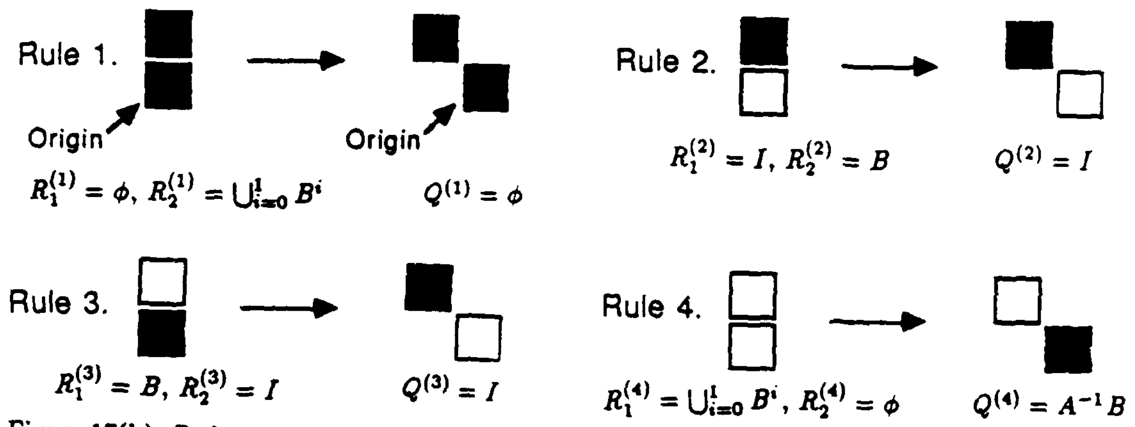

Rule 1.
Origin
Origin
$R_1^{(1)} = \phi$, $R_2^{(1)} = \bigcup_{i=0}^{1} B^i$
$Q^{(1)} = \phi$
Rule 2.
$R_1^{(2)} = I$, $R_2^{(2)} = B$
$Q^{(2)} = I$
Rule 3.
$R_1^{(3)} = B$, $R_2^{(3)} = I$
$Q^{(3)} = I$
Rule 4.
$R_1^{(4)} = \bigcup_{i=0}^{1} B^i$, $R_2^{(4)} = \phi$
$Q^{(4)} = A^{-1}B$

Figure 17(b): Reference image pairs $R^{(i)}$ and reference images $Q^{(i)}$, $i = 1,2,3,4$, used for addition. $Q^{(1)}$ is a null image, Rule 1 is not needed for this simple intensity coding.

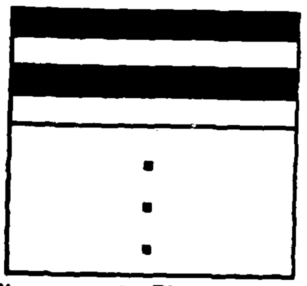


Figure 17(c): The mask M.

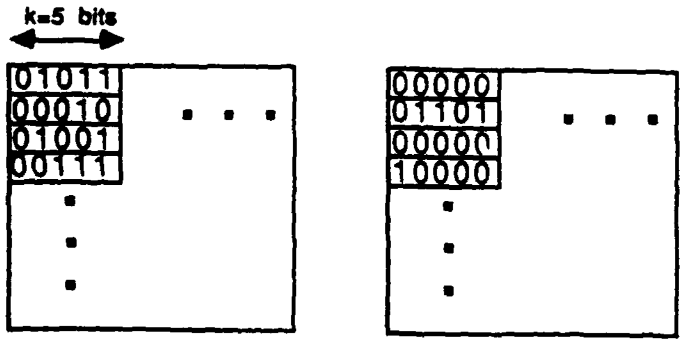


Figure 17(d): An example of parallel addition of binary symbol-coded numbers.

Figure 17: Parallel addition of binary symbol-coded numbers.

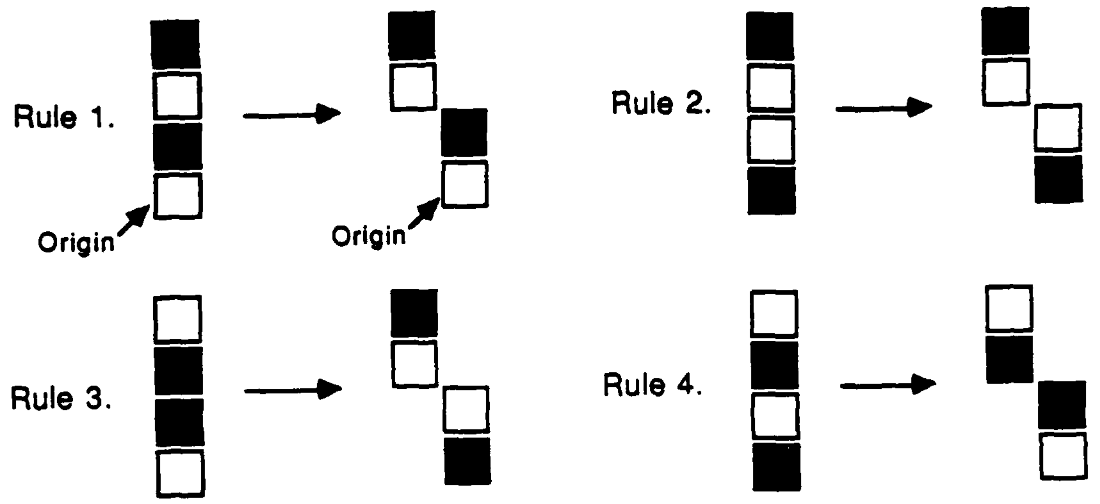


Figure 18(a): Reference image pairs $R^{(i)}$ and reference images $Q^{(i)}$, $i = 1,2,3,4$, used for addition (with dual-rail coding) (adapted from [18] [19]).

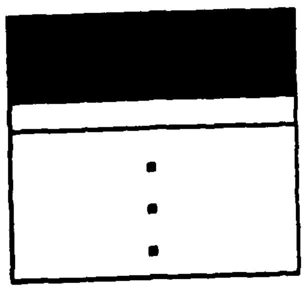


Figure 18(b): The mask M.

Figure 18: Symbolic substitution binary addition with dual-rail coding.

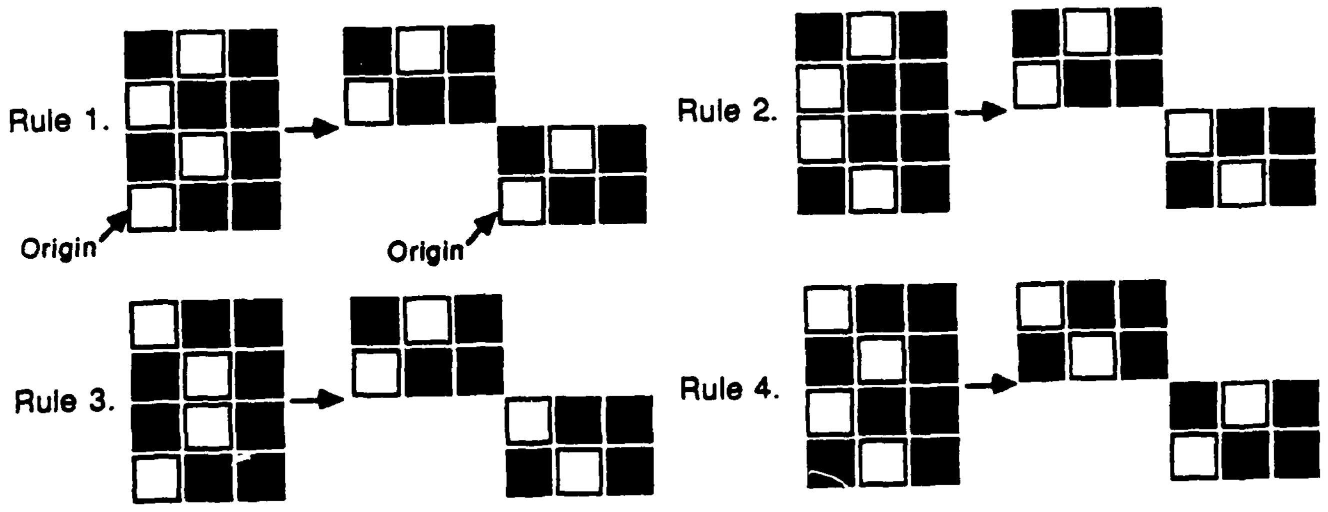


Figure 19: Symbolic substitution binary addition with encoding a bit as six pixels (adapted from [30]).

```
                     10100
                    -10011
Difference bits      00111
 Borrow bits        00011
Difference bits      00001
 Borrow bits        0011
Difference bits      00001
 Borrow bits         000
```

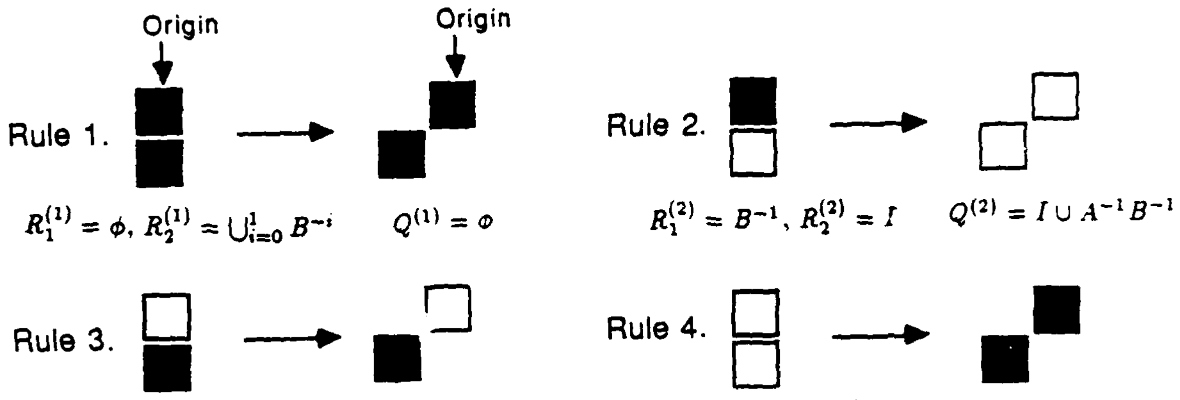
Rule 1. $\substack{0\\0} \rightarrow {}_{0}{}^{0}$

Rule 2. $\substack{0\\1} \rightarrow {}_{1}{}^{1}$

Rule 3. $\substack{1\\0} \rightarrow {}_{0}{}^{1}$

Rule 4. $\substack{1\\1} \rightarrow {}_{0}{}^{0}$

Figure 20(a): Four symbolic substitution rules for subtraction.



$R_1^{(1)} = \phi,\ R_2^{(1)} = \bigcup_{i=0}^{1} B^{-i}$ $\quad Q^{(1)} = \phi$

$R_1^{(2)} = B^{-1},\ R_2^{(2)} = I$ $\quad Q^{(2)} = I \cup A^{-1}B^{-1}$

$R_1^{(3)} = I,\ R_2^{(3)} = B^{-1}$ $\quad Q^{(3)} = I$

$R_1^{(4)} = \bigcup_{i=0}^{1} B^{-i},\ R_2^{(4)} = \phi$ $\quad Q^{(4)} = \phi$

Figure 20(b): Reference image pairs $R^{(i)}$ and reference images $Q^{(i)}$, $i = 1, 2, 3, 4$, used for subtraction. Because $Q^{(1)}$ and $Q^{(4)}$ are null images, Rules 1 and 4 are not needed for simple intensity coding.

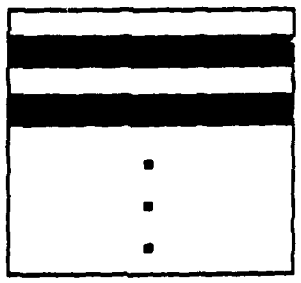


Figure 20(c): The mask M.

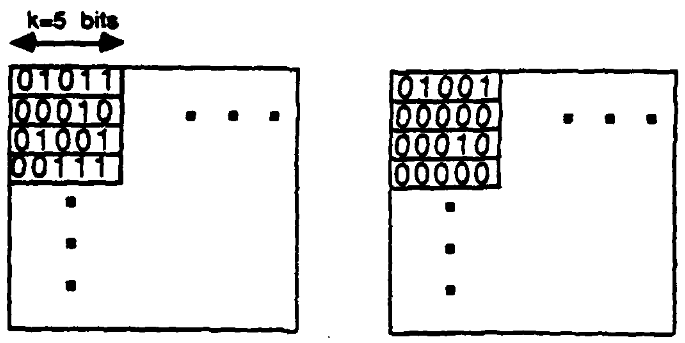


Figure 20(d): An example of parallel subtraction of binary symbol-coded numbers.

Figure 20: Parallel subtraction of binary symbol-coded numbers.

Number Representation	Binary Row-coding	Binary Stack-coding	Symbolic Substitution (simple intensity coding)	Symbolic Substitution (dual-rail coding)
No. of Dilations (or Erosions)	k	0	9(k+1)	8(k+1)
No. of Unions (or Intersections)	4k+3	16k-12	15(k+1)	16(k+1)
No. of Complements	7k+4 *2k+2	20k-13 *7k-5	12(k+1) *3(K+1)	16(k+1) *4(k+1)
Total No. of Parallel Fundamental Operations O	12k+7 *7k+5	36k-25 *23k-17	36(k+1) *27(k+1)	40(k+1) *28(k+1)
No. of Processing Elements P	kN^2	N^2	$2kN^2$	$4kN^2$
Total No. of Computations OxP	$(12k+7)kN^2$ $*(7k+5)kN^2$	$(36k-25)N^2$ $*(22k-16)N^2$	$76k(k+1)N^2$ $*54k(k+1)N^2$	$160k(k+1)N^2$ $*112k(k+1)N^2$
DOCIP Execution Time T	5k+3	17k-13	17(k+1)	18(k+1) or 28(k+1)
PxT	$(5k+3)kN^2$	$(17k-13)N^2$	$34k(k+1)N^2$	$72k(k+1)N^2$ or $112k(k+1)N^2$

* indicates the number of operations when erosion and intersection are also allowed.

Table 1. Complexity of parallel optical binary addition of two NxN arrays of k-bit binary numbers. Each parallel fundamental operation corresponds to P processing elements executing in parallel.

Number Representation	Binary Row-coding	Binary Stack-coding	Symbolic Substitution (simple intensity coding)	Symbolic Substitution (dual-rail coding)
No. of Dilations (or Erosions)	k	0	6(k+1)	8(k+1)
No. of Unions (or Intersections)	4k+3	16k-12	10(k+1)	16(k+1)
No. of Complements	6k+4 *3k+2	22k-18 *11k-8	8(k+1) *2(K+1)	16(k+1) *4(k+1)
Total No. of Parallel Fundamental Operations O	11k+7 *8k+5	43k-33 *33k-26	24(k+1) *18(k+1)	40(k+1) *28(k+1)
No. of Processing Elements P	kN^2	N^2	$2kN^2$	$4kN^2$
Total No. of Computations OxP	$(11k+7)kN^2$ $*(8k+5)kN^2$	$(43k-33)N^2$ $*(33k-26)N^2$	$48k(k+1)N^2$ $*36k(k+1)N^2$	$160k(k+1)N^2$ $*112k(k+1)N^2$
DOCIP Execution Time T	4k+3	20k-17	11(k+1)	18(k+1) or 28(k+1)
PxT	$(4k+3)kN^2$	$(20k-17)N^2$	$22k(k+1)N^2$	$72k(k+1)N^2$ or $112k(k+1)N^2$

* indicates the number of operations when erosion and intersection are also allowed.

Table 2. Complexity of parallel optical binary subtraction of two NxN arrays of k-bit binary numbers.

2.1 Binary Image Algebra

Binary image algebra (BIA) forms the mathematical background for software and hardware systems suitable for optical digital computing. Parallel algorithms for optical cellular logic and symbolic substitution processors can be formalized as compact BIA expressions. BIA also leads to the architectural design of digital optical cellular image processors (DOCIP) which are well-suited to executing the parallel algorithms. The following paper "An Image Algebra Representation of Parallel Optical Binary Arithmetic" submitted to *Applied Optics* summarizes our recent work in this area.

Binary Image Algebra and Optical Cellular Logic Processor Design[†]

K. S. Huang, B. K. Jenkins, and A. A. Sawchuk

Signal and Image Processing Institute
Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089-0272

[o†]**Supported by the Air Force Office of Scientific Research under grant AFOSR-84-0181 and by an IBM graduate fellowship.**

[‡]**Portions of this paper presented at the 1987 IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Machine Intelligence [1].**

[*]**Submitted to Computer Vision, Graphics, and Image Processing.**

Abstract

Is there a simple unified consistent complete theory of parallel binary digital image processing algorithms and architectures? Can this theory suggest new parallel architectures for image processing? Can these architectures be implemented by optical computing techniques? We attempt to answer these questions.

Techniques for digital optical cellular image processing are presented. A binary image algebra (BIA), built from only five elementary images and three fundamental operations, serves as its software and leads to a formal parallel language approach to the design of parallel binary image processing algorithms. Its applications and relationships with other computing theories demonstrate that BIA is a powerful systematic tool for formalizing and analyzing parallel algorithms. Digital optical cellular image processors (DOCIPs), based on cellular automata and cellular logic architectures, serve as its hardware and implement the parallel binary image processing tasks naturally and effectively. An algebraic structure provides a link between the algorithms of BIA and architectures of DOCIP. Optical computing suggests a more ideal and efficient implementation of the DOCIP architectures because of its inherent parallelism and 3-D global free interconnection capabilities. Finally, the instruction set and the programming of the DOCIPs are illustrated.

1 Introduction

In this paper we combine studies of architectures, algorithms, mathematical structures, and optics to show that: 1) an image algebra extending from mathematical morphology [2]-[5] can lead to a formal parallel language approach to the design of image processing algorithms; 2) cellular automata are appropriate models for parallel image processing machines [6][7]; 3) an algebraic structure serves as a framework for both algorithms and architectures of parallel image processing; and 4) the parallel processing and global interconnection advantages of optical computing may be useful in efficiently implementing image algebra with cellular logic architectures.

The purpose of the image algebra approach in this paper is for the development of a programming language for a specific parallel architecture, namely a digital optical cellular image processor (DOCIP). The binary image algebra (BIA) described here is based on a set of three specific fundamental operations. These fundamental operations are the key operations in the instruction set of the DOCIP machine. The BIA provides a decomposition of general operations, including low-level image processing operations, into the three fundamental operations of the instruction set. This decomposition is inherently parallel and provides a direct mapping to the machine architecture.

In this section, we first review previous work on image algebra, cellular automata and cellular logic architectures, then define the algebraic structure and outline the detailed discussion that follows.

Previous Work on Image Algebra

During the past few years, numerous papers have used an algebraic approach to aid in image processing [2]-[5] [8]-[10]. Among them, morphological image algebra has the closest relation to binary image algebra (BIA). Many papers describe either specific theoretical aspects of mathematical morphology or application-specific morphological algorithms [11]-[18]. The applications of mathematical morphology has been fruitful. In this paper we adapt it to provide the following features:

1. A simplified mathematical structure. Mathematical morphology comprises two branches, integral geometry and geometrical probability, plus a few collateral ancestors (harmonic analysis, stochastic processes, algebraic topology) [2]. The mathematical details and formal proofs in morphology are often intricate and involve advanced set theoretic and topological concepts which are not always necessary for engineering applications.

2. A complete algebraic theory. Mathematical morphology defines some algebraic operators and utilizes some algebra. With our adaptation, we would like to answer the following questions:

   - What is the algebraic definition of this mathematical morphology?
   - How powerful is this mathematical morphology?
   - What is the definition of a transformation? Morphological transformations are constrained by four principles [2], here we introduce a complete definition of image transformations.

3. Clarification of its relationship to other areas. We define its relationship to linear system theory, image processing, and common computing techniques including boolean logic, cellular logic, and algebraic structures.

Here we develop a simple unified complete parallel binary image processing theory based on an algebraic structure — binary image algebra (BIA). In BIA, parallel binary image processing algorithms (including parallel numerical computations) can be written as compact algebraic expressions where an algebraic symbol represents an image (not a pixel) or an image operation (not a pixel-wise operation). A complete algebraic system comprises three fundamental operations and five elementary images which can be combined to generate any image in the three fundamental operations for forming any image transformation. (In fact, one can define four elementary images and two fundamental operations that are sufficient: however, in this paper we will not consider them since they are more difficult to use.)

There are other image algebras, each with its own characteristics [8] [9]. Because of our intended application to a highly parallel computing machine with simple processing elements and a reduced instruction set, we utilize a BIA with only three fundamental operations that can implement any binary image transformation. For example, the counting function, which gives the number of pixels having a certain level, is considered a mapping from a picture type of operand to another number type of operand in references [8] and [9]; in BIA numbers are also represented as images [19]. BIA suggests several simple but fast parallel image algorithms and a parallel image processing architecture with a very low cell complexity.

Previous Work on Cellular Logic Architectures

To match BIA parallel algorithms by cellular logic architectures in a transparent way, we characterize a cellular automaton by an algebraic structure as BIA does. The cellular logic computer was first inspired by the writings of von Neumann [20][21] on cellular automata. A sequential process of cellular logic operations is described in Fig. 1. Some review of cellular image processors can be found in Ref. [21]-[25]. Many cellular computers have been constructed previously for implementing cellular logic operations, and some ideas for extending the nearest-neighbor connected cellular logic computers for improving speed and flexibility have been proposed [24]. These architectures include: (1) the cellular string (Fig. 2(a)); (2) the cellular array (Fig. 2(b)); and (3) the cellular hypercube (Fig. 2(c)) and the cellular pyramid (Fig. 2(d)). These three architectures share a common feature in the simplicity and regularity of interconnecting simple processing elements, and represent an interconnection topology in 1-D, 2-D, and 3-D, respectively. The 3-D case is difficult to implement on a planar VLSI chip [24] [26] [27], but may be realizable by a digital optical system [28][29]. Two promising architectures based on digital optical cellular image processors (DOCIPs), DOCIP-array and DOCIP-hypercube, are presented below as a means of implementing BIA effectively.

Definition of Algebraic Structure

An algebraic structure (or algebra) [30]-[32] is a pair (or system) $A = (S, F)$ where

- S is a set, and
- F is a family of operations which are functions:

$$f : S^k \rightarrow S,$$

and k is a finite non-negative integer.

Remark: For any finite non-negative integer k, we define a k-ary operation on S as an operation which is a function $f : S^k \rightarrow S$. Thus, a unary (or 1-ary) operation on S is simply a function on S to S. A binary (or 2-ary) operation on S is a function on S^2 to S. For completeness, we define a nullary (or 0-ary) operation on S to be a particular element of S.

Therefore, the problem to be solved is essentially to find a "good" algebraic structure (S, F) for parallel binary image processing, i.e. to search for S and F, and its "good" hardware implementation.

Outline

Section 2 contains the framework of BIA: subsection 2.1 gives the basic definitions; subsection 2.2 presents two fundamental principles which prove the completeness of BIA.

Section 3 describes some applications of BIA: subsection 3.1 reviews basic properties of images and image transformations, and derives from them some standard image operations; subsection 3.2 gives some useful theorems for properties of image operations, for applications in morphological filtering, shape recognition, "salt" and "pepper" noise removal, size and location verification.

Section 4 discusses the relationship of BIA and other computing theories: subsection 4.1 describes the relationship with boolean logic; subsection 4.2 describes the relationship with symbolic substitution and cellular logic; subsection 4.3 describes the relationship with linear shift invariant system theory, convolution and correlation; subsection 4.4 descibes some standard algebraic structures supported by BIA.

Section 5 contains the implementation of BIA on digital optical cellular image processors (DOCIPs): subsection 5.1 gives the algebraic description of the DOCIPs which have the same algebraic structure as BIA; subsection 5.2 gives the general description of the DOCIPs.

Finally, the programming of the DOCIPs is illustrated in Section 6.

2 Binary Image Algebra (BIA) Fundamentals

The overall philosophy of BIA is:

- *An image, but not a pixel, is an object.* For parallel languages and machines for image processing, images can be considered as primitive variables for simplifying the design.
- *Complex image processing operations can be reduced to simple instructions.* Although image processing operations appear complex, the fundamental interactions and the elementary components in a system are very simple.

Thus, BIA begins by:

1. Defining the universal image as the working space for images and their image transformations.
2. Defining elementary images which can be combined to generate any image.
3. Defining fundamental operations which can be cascaded to form complex operations.

4. Defining image processing/analysis algorithm design as the choice of "good" (or "appropriate") reference images and transformations.

A reference image can be any image and is a generalization of structuring elements in mathematical morphology [2]. Reference images contain some predefined image property (or information); image transformations (or operations) are used for measuring the image property from an input image. Image description, image information extraction or image property measurement is done by using reference images to model or transform the original image to a final state which reveals the desired information or is used to detect the desired properties easily.

Here we give the algebraic structure of BIA first, and then provide definitions and present two fundamental principles which allow us generate any reference image and implement any image transformation. Ideally, BIA may be further generalized to GIA (General Image Algebra) which deals with grey-level and complex-valued images.

2.1 Definitions

Definition of Binary Image Algebra (BIA)

Binary image algebra is an algebra with an image space S, which is the power set of a predefined universal image $P(W)$, and a family F of operations including 3 fundamental operations $(\oplus, \cup, ^{-})$, which are non 0-ary operations, and 5 elementary images $(I, A, A^{-1}, B, B^{-1})$, which are 0-ary operations. Symbolically,

$$\text{BIA} = (P(W); \oplus, \cup, ^{-}, I, A, A^{-1}, B, B^{-1})$$

i.e. $S = P(W)$ and $F = (\oplus, \cup, ^{-}, I, A, A^{-1}, B, B^{-1})$. The image space S and the family F of operations will be derived in the following.

Basic Definitions

In general, a binary digital image is defined as a function f that maps each grid point (x, y) of the picture on an orthogonal coordinate system onto the set composed of two elements: 1 (i.e. white, foreground point or image point) and 0 (i.e black or background point). However, it will be more convenient for our algebra, if we use a set of the coordinates of image points ('1's) to specify an image. In this paper, an image is treated as the set of coordinates of image points (i.e. foreground points or pixels that have value 1). We begin the description of BIA by defining our artificial universe:

Definition 2.1 The Universal Image.
The universal image is the set $W = \{(x, y) \mid x \in Z_n, y \in Z_n\}$, where $Z_n = \{0, \pm1, \pm2, ..., \pm n\}$ and n is a positive integer (Fig. 3).
Remark: "$\in$" means "belongs to". Notice that given n, the universal image defines the domain of our images. In fact, for an image with size larger than $(2n + 1) \times (2n + 1)$ (the size of the universal image), we need to increase the size of the universal image or decompose the tested image into subimages whose sizes are smaller than the size of the universal image. For the reason of simple practice, we only consider the square tessellation of images. To deal with non-square tesselations (such as the hexagonal digitization), we can simply replace the universal image to be the set of grid points corresponding to the new digitization pattern.

Definition 2.2 Image Space.
The image space is the power set (the set of all subsets) of the universal image, i.e. $S = P(W)$.

Definition 2.3 Image.
A set X is an image if and only if X is an element of the image space S, i.e. X is a subimage (subset) of the universal image W. Symbolically,

$$X \text{ is an image} \leftrightarrow X \in S \leftrightarrow X \subset W.$$

Remark: "$\subset$" means "is included in". There exist $2^{(2n+1)\times(2n+1)}$ different images. Three terms related to images are defined:

1. Size (or area) of an image X, denoted as $\#(X)$, is the cardinality (i.e. the number of elements) of the image X.
2. Foreground of an image X, simply denoted as X, is referred to those pixels with value 1.
3. Background of an image X, denoted as the complement $\overline{X}$ (Definition 2.6), is referred to those pixels with value 0.

Once we know the foreground of an image, the background of this image is well defined (since the universal image is given first). Thus, the foreground is sufficient to specify an image.

Definition 2.4 Image Point (Foreground Point).
A point (x, y) is an image point of an image X if and only if (x, y) is an element of the set X.
Remark: The largest image is the universal image W and consists of $(2n + 1) \times (2n + 1)$ image points, i.e. $\#(W) = (2n + 1) \times (2n + 1)$; the smallest image is the null image ϕ (defined as the complement $\phi = \overline{W}$) and has no image points, i.e $\#(\phi) = 0$.

Definition 2.5 Image Transformation.
A transformation T is an image transformation if and only if T is a function mapping from the image space S to the image space S.
Remark: There exist $(2^{(2n+1)\times(2n+1)})^{(2^{(2n+1)\times(2n+1)})}$ image transformations.

Definition 2.6 Three Fundamental Operations.
There are three fundamental operations:

1. Complement of an image X (Fig. 4(a)):

$$\overline{X} = \{(x, y) \mid (x, y) \in W \wedge (x, y) \notin X\}$$

2. Union of two images X and R (Fig. 4(b)):

$$X \cup R = \{(x, y) \mid (x, y) \in X \vee (x, y) \in R\}$$

3. Dilation of two images X and R (Fig. 4(c)):

$$X \oplus R = \begin{cases} \{(x_1 + x_2, y_1 + y_2) \in W \mid (x_1, y_1) \in X, (x_2, y_2) \in R\} & (X \neq \phi) \wedge (R \neq \phi) \\ \phi & \textit{otherwise} \end{cases}$$

Remark: "$\wedge$" means "and", and "$\vee$" means "or". Note that X usually represents an input or data image and R is a reference image. The consideration of null image in the dilation operation is missing in mathematical morphology (where the dilation is defined the union of all translations of X by all image points in R); with this generalization we have a complete theory which is not found in other image algebras because of less demonstration of their capabilities for implementing any image transformation. We can also define other image operations as fundamental operations instead of these three operations. The reason for choosing these three operations is because of their simplicity, simple software design and simple hardware implementation. As shown later, these three operations may be implemented by a 2-D optical gate array with 3-D interconnections.

Definition 2.7 Elementary Images.
These elementary images are constant images, i.e. 0-ary operations. Each elementary image has only one image point. There are 5 elementary images:

1. $I = \{(0,0)\}$ — consisting of an image point at the origin
2. $A = \{(1,0)\}$ — consisting of an image point right of the origin
3. $A^{-1} = \{(-1,0)\}$ — consisting of an image point left of the origin
4. $B = \{(0,1)\}$ — consisting of an image point above the origin
5. $B^{-1} = \{(0,-1)\}$ consisting of an image point below the origin

Remark: In fact, these 5 elementary images could be reduced to 4 elementary images, because $I = A^0 \equiv A \oplus A^{-1} = B^0 \equiv B \oplus B^{-1}$.

Definition 2.8 Reflected Reference Image.
Given a reference image R which is a predefined image for containing some desired image property or image information, its refected image is defined as

$$\check{R} = \{(-x,-y) \mid (x,y) \in R\}.$$

Remark: In many useful cases the reference image R is symmetric, then $\check{R} = R$.

2.2 Two fundamental Principles

Two fundamental principles basically define the binary image algebra (BIA). Before stating these two principles, we give some preliminary results.

Lemma 2.1.

$$\overline{(X \oplus \overline{R}) \cup (\overline{X} \oplus R) \cup \overline{I}} = \begin{cases} I & if X = \check{R} \\ \phi & otherwise \end{cases} \quad \forall X, R \in P(W)$$

where $I = \{(0,0)\}$ is an elementary image, $\check{R}$ is the reflected reference image of R, and "$\forall$" means "for all".

Proof: Appendix A.

Remark: This lemma says that if the image X matches the image $\check{R}$, then the origin (central pixel) of the above output image has value '1', otherwise always '0'.

Theorem 2.1.

Any image transformation $T : P(W) \rightarrow P(W)$ can be expressed as

$$T(X) = \bigcup_{i=1}^{k} \{\overline{(\overline{X} \oplus R_i) \cup (X \oplus \overline{R_i}) \cup \overline{I}} \oplus Q_i\}$$

where $k \leq \#(P(W))$, R_i and Q_i are the reference images used to form any desired image transformation, and

$$\bigcup_{i=1}^{k} R_i \equiv R_1 \cup R_2 \cup \ldots \cup R_k.$$

Proof: Appendix B.

Theorem 2.2.

Any image can be represented as

$$X = \bigcup_{(i,j) \in X} A^i B^j$$

where $A^i B^j \equiv A^i \oplus B^j$,

$A^i \equiv \underbrace{A \oplus A \oplus \ldots \oplus A}_{i} = \{(i,0)\}$ if $i > 0$,

$A^i \equiv \underbrace{A^{-1} \oplus A^{-1} \oplus \ldots \oplus A^{-1}}_{-i} = \{(i,0)\}$ if $i < 0$,

and A, B, A^{-1}, B^{-1} are the elementary images defined in Definition 2.7.

Proof: Appendix C.

Principle 1. Fundamental Principle of Image Transformations

Any image transformation T can be implemented by using appropriate reference images R and the three fundamental operations: 1. Complement $\overline{X}$ of an image X, 2. Union $\cup$ of two images, 3. Dilation $\oplus$ of two images.

Proof: It follows from Theorem 2.1.

In order to use principle 1 efficiently in practice, we invoke principle 2 for the generation of reference images.

Principle 2. Fundamental Principle of Reference Images

Any reference image R can be generated from elementary images $(I, A, A^{-1}, B, B^{-1})$ by using the three fundamental operations.

Proof: It follows from Theorem 2.2.

Therefore, by the above principles, we can represent BIA as:

$$\text{BIA} = (P(W); \oplus, \cup, {}^{-}, I, A, A^{-1}, B, B^{-1}).$$

3 Development of Binary Image Algebra (BIA)

BIA can have many applications in character recognition, industrial inspection, medical image processing, and scientific computation. In this section we first review the basic properties of images and image transformations, define 11 standard operations, and give some special cases of dilation [2]-[5] [33]-[36]. Then we summarize 5 theorems for binary image processing.

This section is primarily a survey of binary image processing algorithms with implementation using BIA fundamental operations. These fundamental operations are so chosen because they form an efficient basis for the instruction set of an optically-based cellular image processor. This survey serves as a description of a parallel language for controlling the processor and how it is compiled into low level instructions. The use of BIA for parallel numerical computation is described in [19].

3.1 Basic Properties of Images and Image Transformations

Definition 3.1 Connectivity in Images

1. 4-neighbor and 8-neighbor:
   An image point (x, y) in an image X can have two types of neighors:
   (a) An image point (i, j) is a 4-neighbor of (x, y)
   $\leftrightarrow (i, j) \in \{(x \pm 1, y), (x, y \pm 1)\}$.
   Remark: $\{(x, y), (x \pm 1, y), (x, y \pm 1)\}$ is called the 4-neighborhood of (x, y) and $N_4 \equiv \{(0,0), (0, \pm 1), (\pm 1, 0)\} = I \cup A \cup A^{-1} \cup B \cup B^{-1}$ (Fig. 5(a)).
   (b) An image point (i, j) is a 8-neighbor of (x, y)
   $\leftrightarrow (i, j) \in \{(x \pm 1, y), (x, y \pm 1), (x \pm 1, y \pm 1)\}$.
   Remark: $\{(x, y), (x \pm 1, y), (x, y \pm 1), (x \pm 1, y \pm 1)\}$ is called the 8-neighborhood of (x, y) and $N_8 \equiv \{(0,0), (0, \pm 1), (\pm 1, 0), (\pm 1, \pm 1)\}$ (Fig. 5(b)).

2. 4-connected and 8-connected:
   (a) Two image points (x, y) and (i, j) of an image X are 4-connected
   $\leftrightarrow$ there exists a sequence of image points $(x, y) = (x_0, y_0), (x_1, y_1), \ldots, (x_m, y_m) = (i, j)$, where (x_k, y_k) is a 4-neighbor of (x_{k-1}, y_{k-1}) and $(x_k, y_k) \in X$, $1 \leq k \leq m$.
   (b) Two image points (x, y) and (i, j) of an image X are 8-connected
   $\leftrightarrow$ there exists a sequence of image points $(x, y) = (x_0, y_0), (x_1, y_1), \ldots, (x_m, y_m) = (i, j)$, where (x_k, y_k) is a 8-neighbor of (x_{k-1}, y_{k-1}) and $(x_k, y_k) \in X$, $1 \leq k \leq m$.

   Remark 1: "4-connected in X" and "8-connected in X" are equivalence relations (reflexive, symmetric and transitive).

   Remark 2: For any image point (x, y) in a non-null image X, the set of (i, j) such that (x, y) and (i, j) are 4-connected (or 8-connected) is called a 4-connected (or 8 connected) component of X. A 4-connected (or 8-connected) component of X is just an equivalence class in X under the equivalence relation — "4-connected (or 8-connected) in X". Thus, a collection of 4-connected (or 8-connected) component of X forms a partition of X, i.e. the set of all 4-connected (or 8-connected) components $\{X_i\}_{i \in I}$ (where I is the index set of connected components) is a family of non-null subimages of X and has the following properties:

(a) $X_i \neq \phi$ for all $i \in I$.

(b) $X_i \cap X_j = \phi$ for all $i \neq j$, $i, j \in I$. ($X_i \cap X_j = \overline{\overline{X_i} \cup \overline{X_j}}$ as defined in Definition 3.3)

(c) $X = \cup_{i \in I} X_i$.

Fig. 6(a) shows a 4-connected component in an image X and Fig. 6(b) s ows an 8-connected component in X.

Remark 3: If an image X has l 4-connected (or 8-connected) components, there are l distinct equivalence classes in X. Each equivalence class X_i can be represented by an image point in X_i. Thus, we may use l distinct image points which belong to l different 4-connected (or 8-connected) components to represent the classes of the image X.

Remark 4: In dealing with connectedness in both X and $\overline{X}$, to avoid the "connectivity paradox" [33], it is preferable to use opposite types of connectedness for X and $\overline{X}$, i.e. if we use "4-connected" for X, then we use "8-connected" for $\overline{X}$, and vice versa.

Remark 5: If any image X is surrounded by a border of 0's, the component of $\overline{X}$ consisting of the points connected to (any one of) these 0's is called the *outside* of X (Fig. 7(a)). If $\overline{X}$ has any other components, they are called *holes* in X (Fig. 7(b)).

For more detailed discussion of geometric properties of images, the reader is referred to [33]-[35]. For equivalence relations, equivalence classes and partitions, please refer to [30]-[32].

Definition 3.2 Basic Properties in Image Transformations

The key properties of image transformations are the following ten basic properties

1. Increasing: An image transformation $T(X)$ is increasing
   $\leftrightarrow (X \subset Y \rightarrow T(X) \subset T(Y))$ for all $X, Y \in P(W)$.

2. Decreasing: An image transformation $T(X)$ is decreasing
   $\leftrightarrow (X \subset Y \rightarrow T(Y) \subset T(X))$ for all $X, Y \in P(W)$.

3. Extensive: An image transformation $T(X)$ is extensive
   $\leftrightarrow X \subset T(X)$ for all $X \in P(W)$.

4. Antiextensive: An image transformation $T(X)$ is antiextensive
   $\leftrightarrow T(X) \subset X$ for all $X \in P(W)$.

5. Idempotent: An image transformation $T(X)$ is idempotent
   $\leftrightarrow T(T(X)) = T(X)$ for all $X \in P(W)$.

6. Shift invariant: An image transformation $T(X)$ is shift invariant
   $\leftrightarrow T(X \oplus P) = T(X) \oplus P$ for all $X, P \in P(W)$ and P is a point image which consists of one and only one image point.
   If an image transformation is not shift invariant, then it is shift variant:
   $T(X \oplus P) \neq T(X) \oplus P$ (in general).

7. Homotopic: An image transformation $T(X)$ is homotopic
$\leftrightarrow$ there exists a one-to-one and onto correspondence between the connected components of X and those of $T(X)$, for all $X \in P(W)$. The same is then true for the holes.

8. Commutative: A binary image operation $\cdot$ is commutative
$\leftrightarrow X \cdot R = R \cdot X$ for all $X, R \in P(W)$.

9. Associative: A binary image operation $\cdot$ is commutative
$\leftrightarrow (X \cdot R) \cdot Q = X \cdot (R \cdot Q)$ for all $X, R, Q \in P(W)$.

10. Distributive: A binary image operation $\cdot$ is distributive over a binary image operation $+$
$\leftrightarrow X \cdot (R + Q) = (X \cdot R) + (X \cdot Q)$ for all $X, R, Q \in P(W)$.

Definition 3.3 Standard Operations

Most standard operations can be derived from the three fundamental operations; eleven common ones follow:

1. Difference of X by R (Fig. 8(a)):

$$X/R = \{(x, y) \in X \mid (x, y) \notin R\} = X \cap \overline{R} = \overline{\overline{X} \cup R}$$

Remark: $\overline{X} = W/X$ where W is the universal image. The difference is an obvious approach to detect defects in the foreground of a tested image.

2. Intersection of two images X and R (Fig. 8(b)):

$$X \cap R = \{(x, y) \mid (x, y) \in X \wedge (x, y) \in R\} = \overline{\overline{X} \cup \overline{R}}$$

Remark: $X \cup R = \overline{\overline{X} \cap \overline{R}}$. If $X \cap R \neq \phi$, then we say that an image X hits (or is joint with) an image R. If $X \cap R = \phi$, then we say that an image X misses (or is disjoint with) an image R

3. Erosion of an image X by a reference image R or foreground template matching of X by R (Fig. 8(c)):

$$X \ominus R = \overline{\overline{X} \oplus \check{R}}$$

Remark: $X \oplus R = \overline{\overline{X} \ominus \check{R}}$, and $R = \check{R}$ when R is symmetic. The erosion of an image X by a reference image R can be thought of as the complement of the dilation of the backgound by the reflection of the reference image R. In general, the erosion of a non-null image X by a non-null reference image R can be used to decrease the size of regions, increase the size of holes, eliminate regions, and break bridges in X; on the contrary, the dilation of a non-null image X by a non-null reference image R can increase the size of regions, decrease or fill in holes and cavities, and bridge gaps in X. Furthermore, the erosion can be interpreted as a foreground template matching where the foreground points of $X \ominus R$ indicates the ocurrences of the foreground template R in X (in this purpose, the size of R usually is much smaller than the size of X).

4. Symmetric Difference of two images (mod 2 image addition or subtraction) (Fig. 8(d)):

$$X \triangle R = (X/R) \cup (R/X) = \overline{\overline{X} \cup R} \cup \overline{\overline{R} \cup X}$$

Remark: The symmetric difference is a commutative operation, and its inverse operation can be defined as itself. In section 4 we show that this operation is the parallel form of boolean EXCLUSIVE-OR. It is an obvious approach to detect defects (including the foreground or background defects) of a tested image.

5. Opening of an image X by a reference image R (Fig. 8(e)):

$$X \circ R = (X \ominus R) \oplus R = \overline{\overline{X} \oplus \check{R}} \oplus R$$

Remark: The opening operation is an erosion followed by a dilaton with the same reference image R. In general, the opening $X \circ R$ with a non-null *reference image* R *reduces the size* of regions and eliminates some image points by removing all features in X which can not contain the reference image R.

6. Closing of an image X by a reference image R (Fig. 8(f)):

$$X \bullet R = (X \oplus R) \ominus R = \overline{\overline{(X \oplus R)} \oplus \check{R}}$$

Remark: The closing operation is a dilation followed by an erosion with the same reference image R. In general, the closing $X \bullet R$ with a non-null reference image R increases the size of regions and eliminates some background points by filling in all background areas that can not contain the reference image R, such as holes and concavities in the image X.

7. Hit or miss transform $\circledast$ of an image X by an image pair $R = (R_1, R_2)$ or template matching of X by R (Fig. 8(g)):

$$X \circledast R = (X \ominus R_1) \cap (\overline{X} \ominus R_2) = \overline{(\overline{X} \oplus \check{R}_1) \cup (X \oplus \check{R}_2)}$$

Remark: The hit or miss transform of an image X by a reference image pair $R = (R_1, R_2)$ is used to match the shape (or template) defined by the reference image pair R where R_1 defines the foreground of the shape and R_2 defines the background of the shape. The key conditions are that the foreground X must match R_1 (i.e. $X \ominus R_1$), while simutaneously the background $\overline{X}$ matches R_2 (i.e. $\overline{X} \ominus R_2$). In order to better define the hit or miss transform and its relationship with conventional boolean logic operations, we start from a pixel-wise boolean comparison to derive the hit or miss transform in shape recognition (Theorem 3.4). Note the similarity of the symmetric difference and the hit or miss transform.

8. Thinning $\circledcirc$ an image X by an image pair $R = (R_1, R_2)$ (Fig. 8(h)):

$$X \circledcirc R = X/(X \circledast R) = \overline{\overline{X} \cup \overline{(\overline{X} \oplus \check{R}_1) \cup (X \oplus \check{R}_2)}}$$

Remark: The thinning operation is antiextensive and decreases the size by removing the central points of the regions which match the reference image pair $R = (R_1, R_2)$.

9. Thickening $\odot$ an image X by an image pair $R = (R_1, R_2)$ (Fig. 8(i)):

$$X \odot R = X \cup (X \circledast R) = X \cup \overline{(\overline{X} \ominus \check{R}_1) \cup (X \ominus \check{R}_2)}$$

Remark: The thickening operation is extensive and increases the size by filling the image points where the regions match the reference image pair $R = (R_1, R_2)$.

10. Sequential operations (e.g. sequential dilation, sequential erosion, sequential thinning etc.): If an image operation $\cdot$ is successively performed with each reference image (or image pairs) in a sequence $(R_\theta) \equiv (R_a, R_b, ..., R_z)$, then we define a sequential image operation

$$X \cdot (R_\theta) = (...((X \cdot R_a) \cdot R_b)... \cdot R_z).$$

Two examples are:

(a) Sequential thinning of an image X by a sequence of image pairs $(R_\theta) \equiv (R_a, R_b, ..., R_z)$:

$$X \circledcirc (R_\theta) = (...((X \circledcirc R_a) \circledcirc R_b)... \circledcirc R_z).$$

Remark: The sequential thinning is powerful in many applications, such as constructing a digital homotopic skeleton of an image X. Skeletonization of an image is an operation that transforms the image to a simplified image, called skeleton, which emphasizes its connectivity. However, a homotopic skeleton cannot be obtained by digitizing an analog skeletonization algorithm; instead, a sequential thinning with a sequence of reference image pairs should be used. Several different algorithms employing different reference image pairs (called masks) have been proposed by several authors [6][36]. Fig. 8(j) shows an example of the skeletonization by a sequential thinning with a sequence of eight reference image pairs proposed by Levialdi et al [36].

(b) Sequential dilation of an image X and a sequence of reference images $(R_\theta) \equiv (R_a, R_b, ..., R_z)$:

$$X \oplus (R_\theta) = (...((X \oplus R_a) \oplus R_b)... \oplus R_z).$$

Remark: Since the dilation is commutative and associative, in practice the dilation $X \oplus R$ with a *large* reference image R is usually implemented as a sequential dilation with a sequence of *small* reference images. For example, if $R = E_1 \oplus E_2 \oplus ... \oplus E_k$, then

$$X \oplus R = (...((X \oplus E_1) \oplus E_2) \oplus ... \oplus E_k);$$

and if $E = E_1 = E_2 = ... = E_k$, then

$$R = E^k \equiv \underbrace{E \oplus E \oplus ... \oplus E}_{k}.$$

11. Conditional operations (e.g. conditional dilation, conditional erosion, conditional thinning etc.):
An image operation $\cdot$ between an image X and a reference image (or image pairs) R performed within a limiting set Y is called a conditional operation and is denoted

$$X \cdot R \mid Y = (X \cdot R) \cap Y = \overline{\overline{X \cdot R} \cup \overline{Y}}$$

Remark: Fig. 8(k) gives an example of the conditional dilation.

3.2 Examples of Special Cases: Translation (Shifting), Expansion, Shrinking, and Projection

Translation (shifting), expansion, shrinking and projection in a direction can by achieved by the dilation (or erosion) in a direct way.

1. Shifting an image X from coordinate (x, y) to coordinate $(x+i, y+j)$ is done by

$$X \oplus \{(i,j)\} = X \ominus \{(-i,-j)\}.$$

   Remark: A point image $\{(i,j)\}$ corresponds to a discrete delta function at $\delta(x-i, y-j)$. Thus, an image function $X(x,y)$ (which corresponds to the image X) convolved with the delta function $\delta(x-i, y-j)$ or correlated with $\delta(x+i, y+j)$ is the same as $X \oplus \{(i,j)\} = X \ominus \{(-i,-j)\}$.

2. Adding a new 8-connected or 4-connected boundary to an image X (i.e. expansion) is done by

$$X \oplus N_4 \text{ or } X \oplus N_8$$
$$\text{where } N_4 \equiv I \cup A \cup A^{-1} \cup B \cup B^{-1} \text{ and } N_8 \equiv \bigcup_{i,j=-1}^{1} A^i B^j.$$

3. Removing the 8-connected or 4-connected boundary of an image X (i.e. shrinking) is done by

$$X \ominus N_4 = \overline{\overline{X} \oplus N_4} \text{ or } X \ominus N_8 = \overline{\overline{X} \oplus N_8}$$
$$\text{where } N_4 \equiv I \cup A \cup A^{-1} \cup B \cup B^{-1} \text{ and } N_8 \equiv \bigcup_{i,j=-1}^{1} A^i B^j.$$

4. Projecting an image X to distance k in a direction θ, i.e. producing a shadow of X where the furthest image point in the shadow in the direction θ is at distance k from the furthest image point in X in the direction θ, this can be achieved by

$$X \oplus \Theta^k$$

   where Θ can be any one of the following:

   - East: $E = I \cup A$, $E^k = \bigcup_{i=0}^{k} A^i$
   - South: $S = I \cup B^{-1}$, $S^k = \bigcup_{i=0}^{k} B^{-i}$
   - West: $W = I \cup A^{-1}$, $W^k = \bigcup_{i=0}^{k} A^{-i}$
   - North: $N = I \cup B$, $N^k = \bigcup_{i=0}^{k} B^i$
   - Southeast: $S_E = I \cup AB^{-1}$, $S_E^k = \bigcup_{i=0}^{k} A^i B^{-i}$
   - Southwest: $S_W = I \cup A^{-1}B^{-1}$, $S_W^k = \bigcup_{i=0}^{k} A^{-i} B^{-i}$
   - Northwest: $N_W = I \cup A^{-1}B$, $N_W^k = \bigcup_{i=0}^{k} A^{-i} B^i$
   - Northeast: $N_E = I \cup AB$, $N_E^k = \bigcup_{i=0}^{k} A^i B^i$
   - Horizontal: $H = \bigcup_{i=-1}^{1} A^i$, $H^k = \bigcup_{i=-k}^{k} A^i$
   - Vertical: $V = \bigcup_{i=-1}^{1} B^i$, $V^k = \bigcup_{i=-k}^{k} B^i$
   - Left-diagonal: $L_D = \bigcup_{i=-1}^{1} A^{-i} B^i$, $L_D^k = \bigcup_{i=-k}^{k} A^{-i} B^i$
   - Right-diagonal: $R_D = \bigcup_{i=-1}^{1} A^i B^i$, $R_D^k = \bigcup_{i=-k}^{k} A^i B^i$

3.3 Theorems for Low Level Vision

Here we summarize five theorems for binary image processing applications. Theorem 3.1 givas basic properties of the BIA fundamental operations and standard operations. Theorems 3.2-3.5 describes the implementation of morphological filters, shape recognition algorithms, "salt" and "pepper" noise removal, and size and location verifications. Those more obvious proofs are omitted for brevity.

Theorem 3.1 Properties of Image Operations

The BIA fundamental operations and standard operations have the properties shown in Table 1(a) and Table 1(b).

Proof: Appendix D gives some of their mathematical expressions which follow form the definitions.

Theorem 3.2 Morphological Filters

Many image transformations are interpreted as morphological filtering [2] or cellular filtering [6]. The major mophological filters are listed in the following:

1. One kind of morphological low pass filter (Fig. 9(a)): to remove high frequencies in the foreground of an image X can be achieved by opening, i.e.

$$X \circ R = (X \ominus R) \oplus R = \overline{\overline{X} \oplus \dot{R}} \oplus R.$$

2. A second kind of morphological low pass filter (Fig. 9(b)): to remove high frequencies in the background of an image X can be achieved by closing, i.e.

$$X \bullet R = (X \oplus R) \ominus R = \overline{\overline{(X \oplus R)} \oplus \dot{R}}.$$

3. A morphological high pass filter (as shown in Fig. 9(c)) which removes low frequencies in the foreground of an image X can be achieved by the difference of X and its opening, i.e.

$$X/(X \circ R) = X/((X \ominus R) \oplus R) = X/(\overline{\overline{X} \oplus \dot{R}} \oplus R) = \overline{\overline{X} \cup (\overline{\overline{X} \oplus \dot{R}} \oplus R)}.$$

4. A morphological band pass filter (as shown in Fig. 9(d)) which removes low frequencies and high frequencies in the foreground of an image X can be achieved by the difference of its opening with a smaller reference image R and its opening with a larger reference image Q, where $R \subset Q$, i.e.

$$\begin{aligned}(X \circ R)/(X \circ Q) &= ((X \ominus R) \oplus R)/((X \ominus Q) \oplus Q) \\ &= ((\overline{\overline{X} \oplus \dot{R}}) \oplus R)/((\overline{\overline{X} \oplus \dot{Q}}) \oplus Q) \\ &= \overline{(\overline{\overline{X} \oplus \dot{R}} \oplus R) \cup (\overline{\overline{X} \oplus \dot{Q}} \oplus Q)}\end{aligned}$$

Theorem 3.3 Shape Recognition (Template Matching)

1. The locations of a shape, that is defined by a non-null reference image R and a non-null reference image (called mask) M (Fig. 10(a)), with $R \subset M \subset W$ (W is the universal image), can be detected by

$$(X \ominus R) \dot{\cap} (\overline{X} \ominus (M/R)) = \overline{(\overline{X} \oplus \dot{R}) \cup (X \oplus (\dot{M}/\dot{R}))} = \overline{(\overline{X} \oplus \dot{R}) \cup (X \oplus \overline{\overline{\dot{M}}} \cup \dot{R})}.$$

   Equivalently, setting $R_1 = R$, $R_2 = M/R$ and redefining a non-null reference image pair $R = (R_1, R_2)$ (Fig. 10(b)) yields the hit or miss transform of X by R:

$$X \circledast R = (X \ominus R_1) \cap (\overline{X} \ominus R_2) = \overline{(\overline{X} \oplus \dot{R}_1) \cup (X \oplus \dot{R}_2)}.$$

2. The locations of a shape, that is defined by a family of non-null reference image pairs $\{R(\theta)\}$ with $\theta \in \Theta$ (Θ is the index set of the family of non-null reference image pairs and $R(\theta) = (R_1(\theta), R_2(\theta))$, can be detected by the union of the hit or miss transform of X by $R(\theta)$:

$$\bigcup_{\theta \in \Theta} X \circledast R(\theta) = \bigcup_{\theta \in \Theta} (X \ominus R_1(\theta)) \cap (\overline{X} \ominus R_2(\theta)) = \bigcup_{\theta \in \Theta} \overline{(\overline{X} \oplus \dot{R}_1(\theta)) \cup (X \oplus \dot{R}_2(\theta))}.$$

Proof: Appendix E.

Theorem 3.4 "Salt" and "Pepper" Noise Removal

1. "Salt" noise removal (isolated image point removal) (Fig. 11(a)): to remove an image point if its 4-connected or 8-connected neighbors are background points (0's) can be achieved by

$$X \odot Q_4 = X \cup \overline{\overline{X} \oplus M_4} \text{ or}$$
$$X \odot Q_8 = X \cup \overline{\overline{X} \oplus M_8}$$

   where $Q_4 = (M_4, \dot{I})$, $Q_8 = (M_8, \dot{I})$, $M_4 = A \cup A^{-1} \cup B \cup B^{-1} = N_4/I$ and $M_8 = N_8/\dot{I}$.

2. "Pepper" noise removal (interior fill) (Fig. 11(b)): to create an image point at a coordinate if its 4-connected or 8-connected neighbors are image points (1's) can be achieved by

$$X \circledcirc R_4 = \overline{\overline{X} \cup \overline{X \oplus M_4}} \text{ or}$$
$$X \circledcirc R_8 = \overline{\overline{X} \cup \overline{X \oplus M_8}}$$

   where $R_4 = (I, M_4)$, $R_8 = (I, M_8)$.

3. "Salt and pepper" noise removal (Fig. 11(c)): to remove noise points, that are completely surrounded with 4-connected neighbors or 8-connected neighbors of the opposite value, can be achieved by

$$(X \odot Q_4)/(X \circledast R_4) = \overline{(X \cup \overline{\overline{X} \oplus M_4}) \cup \overline{(\overline{X} \cup (X \oplus M_4))}} \text{ or}$$
$$(X \odot Q_8)/(X \circledast R_8) = \overline{(X \cup \overline{\overline{X} \oplus M_8}) \cup \overline{(\overline{X} \cup (X \oplus M_8))}}.$$

Proof: Appendix F.

Remark: This theorem demonstrates the fact that many higher level operations (e.g. involving thinning and thickening) can be efficiently :nplemented by the three fundamental operations. Using the same design methodology as the "salt and pepper" noise removal, we can design many similar algorithms, such as spur removal, bridge break, and edge detection (perimeter) etc. For example, the detection of the 4-connected or 8-connected edge of an image X (Fig. 12) can be achieved by

$$X/(X \ominus N_8) = \overline{\overline{X} \cup (\overline{X} \oplus N_8)} \text{ or}$$
$$X/(X \ominus N_4) = \overline{\overline{X} \cup (\overline{X} \oplus N_4)}.$$

Theorem 3.5 Size and Location Verification

The locations in an image X of the regions including the reference image R and included in the reference image Q, where $R \subset Q$, can be detected by

$$S((X \ominus R)/((X \ominus Q) \oplus Q))) = S(\overline{(\overline{X} \oplus \check{R}) \cup (\overline{\overline{X} \oplus \check{Q}} \oplus Q)}).$$

where $S(\cdot)$ means the homotopic skeletonization. (An example is given in Fig. 13.)

Proof: Appendix G.

The above theorems serve as the typical rules for morphological image processing. In fact, there are many ways to analyze the shapes and sizes of an image only using by the three fundamental operations. As another example: comparing an image X with its convex hull $C(X)$ [34] is a useful technique to analyze shape. If there is only one object or objects separated by distances greater than their own diameters in the image X. then its convex hull is the intersection of projections (Fig. 14(a)):

$$\bigcap_{i=1}^{4}(X \oplus \Theta_i^k)$$

where Θ_i, $i = 1,2,3,4$, are H, V, R_D, L_D (defined in Definition 3.4), and k should be greater than the longest radius of objects in X. Then the difference of the convex hull and the image $C(X)/X$ indicates how many concavities the image X has and what their individual shapes and sizes are. Fig. 14(b) illustrates an example.

4 Relationship to Other Computing Theories

4.1 Relationship to Boolean Logic

BIA can implement any boolean logic operation on binary images. It is also obvious that BIA fundamental operations can be implemented by a boolean logic gate array with interconnections. The following straight-forward correspondance can be drawn between the BIA operations and boolean logic operations:

BIA Operations	*Boolean Logic Operations*
1. Complement	NOT
2. Union	OR
3. Dilation	Multiple-input OR
4. Intersection	AND
5. Erosion	Multiple-input AND
6. Symmetric Difference	EXCLUSIVE-OR

Note that the inputs of OR and AND (corresponding to union and intersection) come from two different images. The multiple inputs of OR and AND (corresponding to dilation and union) come from the same image while the other operand image R only determines the number and location of input pixel values. A complete logical set is able to implement any boolean logic function; it consists of at least one of the following sets: NOT and OR; NOT and AND; NAND; NOR. In BIA, in order to implement any image transformation, we need a complete system of pixel-wise logic operations and we also need a translational type of operation (such as translation, dilation, erosion, convolution and correlation etc.) to allow the global information extraction in an image or the information exchange between pixels of the same images. In order to have a 2-D compact parallel form of image processing algorithms whose variables are whole images, we define the parallel form of those corresponding boolean logic operations as BIA operations. In fact, there are two boolean algebras, $(P(W); \cup, \cap, ^{-}, \phi, W)$ and $(P(W); \Delta, \cap, ^{-}, \phi, W)$, supported by BIA also (subsection 4.4). We can define several possible sets of fundamental operations for implementing any image transformation, such as a parallel form of NOR (or NAND or (NOT and OR) or (NOT and AND)) and a translational-type operation (e.g. translation, dilation, erosion, convolution, and correlation etc). The reason why we choose the complement, the union and the *dilation* as the three fundamental operations is:

- Nice mathematical properties: The dilation is commutative, associative, and distributive with the union; but the erosion has no such properties.
- Simple hardware implementation: These three operations are easily implemented by the 2-D gate array and 3-D interconnection technique.
- Simple software design: These three operations are inherently parallel and frequently used operations. Algorithms can be written as compact formulas which easily become very efficient fast parallel algorithms by simply applying the fundamental operations and removing the data dependencies.

Comparing BIA with the conventional boolean expressions for logic functions, the major advantages of BIA are summarized in the following:

- BIA operations are inherently parallel, but boolean logic operations are serial.
- BIA operations include parallel information transferring capabilities which is missing in boolean logic operations.
- Algorithms in BIA are written as compact algebraic formulas whose variables are whole images, while a typical image processing algorithm is very difficult to write in a compact precise boolean logic expression.
- BIA has pictorial physical meaning, while boolean expressions provide little physical feeling for parallel image processing algorithms.

4.2 Relationship to Symbolic Substitution and Cellular Logic

Symbolic substitution is a means of performing parallel digital computions and can be used to implement boolean logic, binary arithmetic, cellular logic and Turing machines [37][38]. It involves two steps: 1) recognizing all the locations of a certain spatial pattern within the 2-D input data, and 2) substituting a new replacement-pattern wherever the search-pattern was recognized. BIA can be used to realize a symbolic substitution rule as follows:

$$(X \circledast R) \oplus Q = \overline{(\overline{X} \ominus \acute{R}_1) \cup (X \ni \acute{R}_2)} \oplus Q$$

where X is the 2-D input data, $R = (R_1, R_2)$ is the reference image pair corresponding to the search-pattern (R_1 and R_2 define the foreground and the background of the search-pattern respectively), $\acute{R}$ defines a reflected reference image given by $\acute{R} = \{(-x, -y) \mid (x, y) \in R\}$, and Q is the reference image corresponding to the replacement-pattern. Thus, symbolic substitution rules are particular BIA image transformations having the above form; and BIA represents a general complete systematic mathematical tool for formalizing the symbolic substitution algorithms.

Cellular logic architectures have been briefly reviewed in section 1. A cellular logic operation transforms an array of data into a new array of data where each element in the new array has a value determined only by the corresponding element in the original array along with the values of its neighbors (Fig. 1). In BIA, an image transformation can be writtern as a polynominal of reference images (Theorem 2.1) where the reference images can have arbitrary large size. In terms of cellular logic, the reference image essentially defines the neighborhood of a cell where the neighborhood can be very large and not just nearest 4- or 8-neighborhood in conventional cellular logic. Thus, cellular logic operations are also particular cases of image transformations with small local reference images, and BIA also serve as a systematic mathematical tool for formalizing cellular logic.

Because of existing hardware interconnection limitations, it is difficult and costly to implement an image transformation with a large reference image in one clock cycle. However, the conventional nearest-neighbor connected cellular arrays have poor communication capabilities. To improve this, we develop the DOCIP-hypercube architecure in section 5, which combines features of conventional nearest-neighbor connected cellular logic architectures and conventional hypercube architectures for implementing BIA effectively.

In summary, BIA provides a systematic mathematical formalism for both symbolic substitution and cellular logic. The applications of symbolic substitution and cellular logic can be accomplished by BIA; on the other hand, generalized cellular logic architectures are good candidates for implementing BIA.

4.3 Relationship to Linear Shift Invariant Systems, Convolution, and Correlation

It is well known that the theory of linear shift-invariant (LSI) systems plays a key role in conventional signal (including image) and system analysis [39][40]. It is very natural that we like to ask what the relation between BIA and LSI system theory is. A system is defined as a transformation or mapping from a set of input functions into a set of output functions, and a two dimensional discrete LSI system is defined as a system which obeys two properties:

- Linearity: $T[ax(i,j) + bz(i,j)] = aT[x(i,j)] + bT[z(i,j)]$ for arbitary constants a and b;

- Shift-invariance: $y(i,j) = T[x(i,j)] \to y(i-k, j-l) = T[x(i-k, j-l)]$.

A linear system can be completely characterized by its unit-impulse response $r(i,j;k,l) = T[\delta(i-k, j-l)]$. In an LSI system the unit-impulse response is simply $r(i,j;k,l) = r(i-k, j-l)$, and the output of an LSI system with input $x(i,j)$ and unit-impulse response $r(i,j)$ is the convolution of $x(i,j)$ and $r(i,j)$, denoted by

$$x(i,j) * r(i,j) = \sum_{k,l=-\infty}^{\infty} x(k,l)r(i-k, j-l).$$

Now, let us consider only binary images. In terms of the set notation, an image $X = \{(i,j) \mid x(i,j) = 1\}$ corresponds to function $x(i,j)$. If we assume $r(i,j) = 1$ at and only at n points which corresponds to an image R with n image points, then the convolution of $x(i,j)$ and $r(i,j)$ with a threshold $t = 0$ is

$$\begin{aligned}
X * R\,|_{t=0} &= \{(i,j) \mid \sum_{k,l} x(k,l)r(i-k, j-l) > 0\} \\
&= \{(i+k, j+l) \mid \sum_{k,l} x(k,l)r(i,j) > 0\} \\
&= \{(i+k, j+l) \mid x(k,l)r(i,j) > 0\} \\
&= \{(i+k, j+l) \mid (i,j) \in X, (k,l) \in R\} \\
&= X \oplus R
\end{aligned}$$

where the ouput of the threshold is defined as 1 if $x(i,j) * r(i,j) > 0$, and is 0 otherwise; and the universal image, as before, contains all image points (i,j), (k,l), and $(i+k, j+l)$. This means that the dilation $X \oplus R$ is the same as adding a threshold $t = 0$ to the convolution sum. The reference image plays a role similar to that of the unit impluse response in the binary image system. Similarly the erosion $X \ominus R$ is the same as the convolution $x(i,j) * r(-i,-j)$ followed by the threshold $t = n-1$.

Correlators have been used in pattern recognition for a long time [41]. Correlation is strongly related to convolution: convolution involves folding, shifting and summing; correlation involves shifting and summing without folding. Therefore,

$$X \oplus R = X * R\,|_{t=0} = X \diamond \check{R}\,|_{t=0}$$

$$X \ominus R = X * \check{R}\,|_{t=n-1} = X \diamond R\,|_{t=n-1}$$

where $*$ means convolution, $\diamond$ means correlation, and $\check{R}$ means the reflected image of R.

Furthermore, although the three fundamental operations of BIA are nonlinear, with appropriate number representations they are able to implement parallel numerical and linear operations too. Also, BIA can implement both shift invariant and shift variant image transformations.

4.4 Some Standard Algebraic Structures

Some algebraic structures supported by BIA are in the following:

1. $(P(W); \oplus)$ is a semigroup.
2. $(P(W); \oplus, I)$ is a monoid.
3. $(P(W); \triangle, \phi, \triangle)$ is an abelian group.
4. $(P(W); \cup, \cap, ^{-}, \phi, W)$ and $(P(W); \triangle, \cap, ^{-}, \phi, W)$ are Boolean algebras.
5. $(P(W); \subset)$ is a poset (partially ordered set).
6. $(P(W); \cup, \cap, \subset)$ is a complete lattice.

Proof: (1) A semigroup is a set with an associative binary operation [30]-[32]. By Theorem 3.1, the dilation $\oplus$ is associative for all images in $P(W)$.

(2) A monoid is a semigroup with an identity [30]-[32]. By Appendix D, the dilation has an identity $I = \{(0,0)\}$. Note that $(P(W); \ominus)$ is neither a semigroup nor a monoid.

(3) A group is a monoid in which every element has an inverse. An abelian group is a group in which the operation is commutative [30]-[32]. By the definition of symmetric difference (mod 2 image addition), it can be easily verified that its identity is ϕ and its inverse operation (mod 2 image subtraction) is itself.

(4) A boolean algebra is a set with operations $\vee, \wedge, ^{-}, 0$ and 1 satisfying: 1. $a \vee b = b \vee a$, $a \wedge b = b \wedge a$ (commutativity); 2. $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$, $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ (associativity); 3. $a \vee 0 = a$ (universal bound); 4. $a \wedge 1 = a$ (universal bound); 5, $a \vee \bar{a} = 1$, $a \wedge \bar{a} = 0$ (complementarity) [30]-[32]. By Appendix D, $(P(W); \cup, \cap, ^{-}, \phi, W)$ and $(P(W); \triangle, \cap, ^{-}, \phi, W)$ are Boolean algebras.

(5) A poset is a set with a relation satisfying: 1. the reflexivity; 2. the antisymmetry; and 3. the transitivity [30]-[32]. The relation $\subset$ satisfies these three conditions: 1. $X \subset X$ for all $X \in P(W)$; 2. if $X \subset R$ and $R \subset X$, then $X = R$; and 3. if $X \subset R$ and $R \subset Q$, then $X \subset Q$.

(6) A complete lattice is a poset $(S; \leq)$ in which every subset of S has a *sup* (the least upper bound) and an *inf* (the greatest lower bound) [30]-[32]. In the algebra $(P(W); \cup, \cap, \subset)$, given any subset of $P(W)$, say $\{X(\theta) \mid \theta \in \Theta\}$ (Θ is the index set of the elements in this subset of $P(W)$), we have

$$sup = \bigcup_{\theta \in \Theta} X(\theta)$$

$$inf = \bigcap_{\theta \in \Theta} X(\theta).$$

Thus, several standard algebraic structures and their properties can be directly implemented and used in BIA.

5 Implementation on Optical Cellular Logic Processors

To map algorithms into architectures, we first use an algebraic approach for describing a cellular image processor. Then we design the digital optical cellular image processors (DOCIPs) and their optical implementation. Figures 15 and 16 show an optic... concept for the DOCIP implementation. The optical system can realize an array of cells by a spatially parallel 2-D array of optical binary gates and performs interconnections of these gates by an optical hologram. The DOCIPs are:

- The DOCIP-array (Fig. 15), a cellular array processor, uses optical parallelism to map an inherently 2-D parallel image data structure to a 2-D nearest-neighbor connected cellular computer in a simple and direct way. Its performance is primarily limited by its $O(1)$ interconnectivity.
- The DOCIP-hypercube (Fig. 16), a two-dimensional cellular hypercube, uses optical parallelism and the 3-D global interconnection capabilities of optics to implement a hypercube interconnection mechanism.

Here, the two-dimensional cellular hypercube is used to match the structure of a two-dimensional image and further improve the communication ability of a cellular array. Ideally, a conventional hypercube (Fig. 17) increases the interconnectivity to $O(logN)$ for N computation cells; however, when laid out in two-dimensional space, its interconnection patterns are not space invariant; such spatial invariance is desirable for image processing and for simple implementation in optical hardware. To include this, we increase the interconnections to make a two dimensional cellular hypercube (Fig. 18). The cellular hypercube introduces a symmetrical positive and negative index so that each cell is connected with cells having a *relative* one bit difference in coordinate label in positive or negative x and y directions; the numerical difference of addresses of connected cells is nonzero in at most 1 bit [42].

5.1 Algebraic Description

Having defined cellular automata and the implementation requirements of BIA, we describe the DOCIP in an algebraic way:

Definition of Cellular Automata

A cellular automaton is an algebra $A = (S; F, N_c)$ where S is the state space which is a set of states, F is a family of transition functions, and N_c is the neighborhood configuration.

Constraints of Implementing BIA:

1. $S \supset P(W)$
2. $F \supset \{\oplus, \cup, ^{-}\}$
3. $N_c \supset I \cup A \cup A^{-1} \cup B \cup B^{-1}$ (or $N_c \supset A \cup A^{-1} \cup B \cup B^{-1}$)

where "$\supset$" means "contains".

Thus, in terms of cellular automata, the DOCIPs have to satisfy the above constraints for realizing BIA. For storing input images and temporary results in a more flexible way, the DOCIPs utilize three memory modules and share the same algebraic structure (except the neighborhood configuration):

$$DOCIP = (P(W \times W \times W); \oplus, \cup, ^{-}, N_c)$$

where "$\times$" denotes cross product and N_c can be one of the following 4 types:

1. DOCIP-array4: each cell connects with its four nearest neighbors and itself, i.e.

$$N_{array4} = I \cup A \cup A^{-1} \cup B \cup B^{-1}.$$

2. DOCIP-array8: each cell connects with its eight nearest neighbors and itself, i.e.

$$N_{array8} = \bigcup_{i,j=-1}^{1} A^i B^j.$$

3. DOCIP-hypercube4: each cell connects with those cells in the 4 directions at distances $1, 2, 4, 8, ..., 2^k$ from itself, i.e.

$$N_{hypercube4} = \bigcup_{i=o,\pm1,\pm2,...,\pm2^k} (A^i \cup B^i)$$

where k is sufficiently large for the connections to traverse the entire array of cells.

4. DOCIP-hypercube8: each cell connects with those cells in the 8 directions at distances $1, 2, 4, 8, ..., 2^k$ from itself, i.e.

$$N_{hypercube8} = \bigcup_{i=o,\pm1,\pm2,...,\pm2^k} (A^i \cup B^i \cup A^i B^i \cup A^i B^{-i})$$

5.2 General Description

From the above algebraic description, the DOCIPs have the same algebraic structure and differ only in their neighborhood configurations N_c. Thus, they share the same architecture as shown in Fig. 19, but have different configurations of the reference images E_i depending on the optical interconnection network which defines the neighborhood. In practical applications, a *larger* reference image R can be generated from a set of *smaller* reference image(s) E_i by a "sequential dilation". If it is possible to decompose R into a sequence $R = E_1 \oplus E_2 \oplus ... \oplus E_k$, then

$$X \oplus R = (...((X \oplus E_1) \oplus E_2) \oplus ... \oplus E_k).$$

This decomposition may not exist, in which case R can always be decomposed as $R = R_1 \cup R_2 \cup ... \cup R_k$, and then

$$X \oplus R = (X \oplus R_1) \cup (X \oplus R_2) \cup ... \cup (X \oplus R_k)$$

where each R_j can be composed from the smaller reference images E_i.

Basically, the proposed DOCIP as shown in Fig. 19 is a cellular SIMD machine and consists of an array of cells or processing elements (PEs) under the supervision of a control unit. The control unit includes a clock, a program counter, a test and branch module for feedback control, and an instruction decoder for storing instructions and decoding them to supervise cells. The array of cells includes a $1 \times 3 \times N^2$ bit destination selector, three $N \times N \times 1$ bit memories for storing images, a memory selector, and a dilation unit.

The DOCIP shown in Fig. 19 operates as follows: (1) a binary image ($N \times N$ matrix) is selected by the destination selector and then stored in any memory as the instruction specifies; (2) after storing the images (1 to 3 $N \times N$ matrices), these images and their complemented versions are piped into the next stage, which forms the union of any combination of images; (3) the result is sent to a dilation where the reference image specified by the instruction is used to control the type of dilation; (4) finally, the dilated image can be output, tested for program control, or fed back to step (1) by the address field of the instruction.

The entire system can be realized by an optical gate array with optical 3-D interconnections [25]-[28]. It should be noted that current optical technology has implemented only arrays of moderately large numbers of gates (500 × 500) at very slow (∼ms) switching speeds, and alternatively, arrays of small numbers of gates (2 × 2 to 6 × 6) at fast switching speeds (0.1μs - 50ps) [43][44]. Current ongoing research in a number of laboratories looks promising in eventually providing the needed arrays of large numbers of gates with reasonably fast switching speeds. Alternatively, control of the DOCIP can be easily realized by using an electronic host instead of the optical control unit, since control of SIMD systems is primarily a serial process. The tradeoff is a possible inefficiency in the interfaces between electronic and optical units. Because of this, the all-optical approach may be preferable in the long term. To efficiently utilize optical gates, they can be interconnected with a 2-D optical multiplexing technique in which a common controllable mask is used for all cells. The optical multiplexing technique has following advantages: 1) the DOCIP will no longer require the broadcasting of instructions from the control unit — instead all cells fan their outputs into a commmon controlling mask pixel; 2) it will reduce the number of gates; and 3) each cell has a simple structure — essentially containing only a 3-bit memory with inverting and non-inverting outputs, and a multiple-input OR gate for dilation.

To avoid the well-known drawbacks of conventional computers based on von Neumann principles [23][38], the machine in Fig. 19 has one instruction which implements the three fundamental operations of BIA along with fetch and store. This design uses the parallelism of optics to simultaneously execute instructions involving all N^2 picture elements.

This single instruction has the following format:

$$(s_1, s_2, ..., s_6, n_1, n_2, ..., n_k, d_1, d_2, d_3, j_1, j_2, a_1, a_2, ..., a_l, b_1, b_2, ..., b_l)$$

where k is determined by the chosen neighborhood configuration N_c. The DOCIP-array requires $k = 5$ or $k = 9$ bits for controlling reference image R at a clock cycle and the DOCIP-hypercube requires $k = O(logN)$ for N cells, and l defines the maximum length of a program: 2^l. The functions of these $11 + k + 2l$ instruction codes are the following:

- $s_1, s_2, ..., s_6$ are used to select the output from the memory elements;
- $n_1, n_2, ..., n_k$ are used to control the neighborhood mask, i.e. to supply the reference image;
- d_1, d_2, and d_3 are used to select the destination memory for storing the image;
- j_1 and j_2 are used to flag an absolute jump or conditional jump;
- $a_1, a_2, ..., a_l$ are the address for jump; and
- $b_1, b_2, ..., b_l$ are the address of the instruction.

Order of magnitude execution times for image processing on the DOCIP machines and on the conventional-array processors are compared in Table 2. In contrast with the DOCIP-array, the DOCIP-hypercube increases the interconnection complexity to $O(logN)$, but is able to perform many global operations in $O(logN)$ time. Comparing with the conventional-array processors having serial or N-parallel input/output, the DOCIP-array will have the same order of performance in local and global operations but will be improved in input/output performance, and in principle could be as low as $O(1)$ in I/O operations. The DOCIP-hypercube will not only be improved in input/output performances but also in global operations. With external memory, it can also be demonstrated to be general purpose in the sense of the ability of simulating any Turing machine. One important feature in the design of the DOCIP-array and DOCIP-hypercube is that optical 3-D free interconnection capabilities can be used to reduce the cell hardware requirements as well as solve the global connection and I/O problems which are difficult to solve by planar VLSI technology.

6 A Programming Example — Size Verification

BIA and DOCIP architectures can have many applications in character recognition, industrial inspection, medical and scientific research. Since BIA is able to implement morphological operations efficiently, the DOCIP machines can efficiently analyze the shape and connectivity of regions as well as measure their size; they also have the potential to accomplish any image transformation. Here we illustrate the programming of the DOCIP machines by a simple size verification algorithm:

- **Problem:** Given an input image X with 31×31 pixels (Fig. 20) which contains some square objects X_i, we want to preserve those square objects X_i which satisfy the following condition:

  $$\text{size of } R \leq \text{size of } X_i < \text{size of } Q$$

  where R and Q are reference images as shown in Fig. 21. Other objects will be eliminated in the output image Y. The expected output image Y is shown in Fig. 22.

- **Algebraic expression for the size verification using band pass morphological filtering (Theorem 3.2):**

  $$\overline{(\overline{\overline{\overline{X} \oplus R} \oplus R}) \cup (\overline{\overline{X} \oplus Q} \oplus Q)}$$

  where $R = \check{R}$ and $Q = \check{Q}$ in this special example.

- **Algorithm for the DOCIP-array8:**

  $$\overline{(\overline{\overline{\overline{X} \oplus E^3} \oplus E^3}) \cup (\overline{\overline{X} \oplus E^4} \oplus E^4)}$$

  where E (Fig. 23) is the allowed reference image with the maximum size at a clock cycle in the DOCIP-array8, the reference images $R = E^3 = E \oplus E \oplus E$ and $Q = E^4 = E \oplus E \oplus E \oplus E = R \oplus E$.

  The DOCIP-array8 requires 13 steps to complete this algorithm, its program (instructions) is in the following:
  Assume start with $X \rightarrow M_1$ (X stored in Memory1)
  1. $\overline{M_1} \oplus E \rightarrow M_2$ $(= \overline{X} \oplus E)$
  2. $M_2 \oplus E \rightarrow M_2$ $(= \overline{X} \oplus E^2)$
  3. $M_2 \oplus E \rightarrow M_2$ $(= \overline{X} \oplus E^3)$
  4. $M_2 \oplus E \rightarrow M_3$ $(= \overline{X} \oplus E^4)$
  5. $\overline{M_2} \oplus E \rightarrow M_2$ $(= \overline{\overline{X} \oplus E^3} \oplus E)$

6. $M_2 \oplus E \rightarrow M_2$ $(= \overline{\overline{X} \oplus E^3} \oplus E^2)$
7. $M_2 \oplus E \rightarrow M_2$ $(= \overline{\overline{X} \oplus E^3} \oplus E^3)$
8. $\overline{M_3} \oplus E \rightarrow M_3$ $(= \overline{\overline{X} \oplus E^4} \oplus E)$
9. $M_3 \oplus E \rightarrow M_3$ $(= \overline{\overline{X} \oplus E^4} \oplus E^2)$
10. $M_3 \oplus E \rightarrow M_3$ $(= \overline{\overline{X} \oplus E^4} \oplus E^3)$
11. $M_3 \oplus E \rightarrow M_3$ $(= \overline{\overline{X} \oplus E^4} \oplus E^4)$
12. $\overline{M_2} \cup M_3 \rightarrow M_3$ $(= (\overline{\overline{\overline{X} \oplus E^3} \oplus E^3}) \cup (\overline{\overline{X} \oplus E^4} \oplus E^4))$
13. End with $\overline{M_3} \rightarrow Y$ $(= \overline{(\overline{\overline{\overline{X} \oplus E^3} \oplus E^3}) \cup (\overline{\overline{X} \oplus E^4} \oplus E^4)})$

- Algorithm for the DOCIP-hypercube8:

$$\overline{(\overline{\overline{\overline{X} \oplus P \oplus E} \oplus P \oplus E}) \cup (\overline{\overline{X} \oplus P \oplus E^2} \oplus P \oplus E^2)}$$

where P (Fig. 24) and E (Fig. 23) are allowed reference images at a clock cycle in the DOCIP-hypercube8, the reference images $R = E^3 = P \oplus E$ and $Q = E^4 = P \oplus E^2 = R \oplus E$.

The DOCIP-hypercube8 requires 10 steps to complete this algorithm, its program (instructions) is shown in the following:
Assume start with $X \rightarrow M_1$ (X stored in Memory1)
1. $\overline{M_1} \oplus P \rightarrow M_2$ $(= \overline{X} \oplus P)$
2. $M_2 \oplus E \rightarrow M_2$ $(= \overline{X} \oplus P \oplus E)$
3. $M_2 \oplus E \rightarrow M_3$ $(= \overline{X} \oplus P \oplus E^2)$
4. $\overline{M_2} \oplus P \rightarrow M_2$ $(= \overline{\overline{X} \oplus P \oplus E^2} \oplus P)$
5. $M_2 \oplus E \rightarrow M_2$ $(= \overline{\overline{X} \oplus P \oplus E^2} \oplus P \oplus E)$
6. $\overline{M_3} \oplus P \rightarrow M_3$ $(= \overline{\overline{X} \oplus P \oplus E^2} \oplus P)$
7. $M_3 \oplus E \rightarrow M_3$ $(= \overline{\overline{X} \oplus P \oplus E^2} \oplus P \oplus E)$
8. $M_3 \oplus E \rightarrow M_3$ $(= \overline{\overline{X} \oplus P \oplus E^2} \oplus P \oplus E^2)$
9. $\overline{M_2} \cup M_3 \rightarrow M_3$ $(= (\overline{\overline{\overline{X} \oplus P \oplus E} \oplus P \oplus E}) \cup (\overline{\overline{X} \oplus P \oplus E^2} \oplus P \oplus E^2))$
10. End with $\overline{M_3} \rightarrow Y$ $(= \overline{(\overline{\overline{\overline{X} \oplus P \oplus E} \oplus E}) \cup (\overline{\overline{X} \oplus P \oplus E^2} \oplus P \oplus E^2)})$

The above programs can be translated into the machine instruction codes directly. If we want to detect the geometric centers (locations) of the desired objects, then we can use a sequential thinning to achieve the homotopic skeleton (Theorem 3.5) (Fig. 25).

7 Conclusions

We have summarized digital optical cellular image processing, including binary image algebra (BIA) and the DOCIP architectures. BIA suggests an unified theory of parallel binary image processing for developing parallel algorithms/languages and can be generalized to grey-level images. Applications of BIA in binary image processing are illustrated. The DOCIP architectures, especially the DOCIP-hypercube, utilize the parallel communication and global interconnection capabilities of optics for avoiding communication bottlenecks and matching BIA parallel algorithms efficiently. A size verification algorithm is used to demonstrate the programming of these

1-instruction DOCIP machines. Overall, BIA is a simple, precise and complete algebraic theory of binary images; the DOCIP machines have simple organization, low cell complexity and potentially fast processing ability.

Ackowledgements

The authors would like to thank T. C. Strand, P. Chavel, E. K. Blum, and R Chellappa for their contributions to this research.

Appendix A

Proof of Lemma 2.1:

We start with the case of $X = \check{R}$ and then the case of $X \neq \check{R}$.

Case 1: $X = \check{R}$, i.e. $R = \check{X}$.

We want to prove

$$\overline{(\overline{X} \oplus \check{X}) \cup (X \oplus \overline{\check{X}}) \cup \overline{I}} = I \leftrightarrow \overline{(\overline{X} \oplus \check{X}) \cup (X \oplus \overline{\check{X}})} \cap I = I.$$

1. Claim $I \subset \overline{(\overline{X} \oplus \check{X}) \cup (X \oplus \overline{\check{X}})} \cap I$

$\leftrightarrow (0,0) \in \overline{(\overline{X} \oplus \check{X}) \cup (X \oplus \overline{\check{X}})}$

$\leftrightarrow (0,0) \notin (\overline{X} \oplus \check{X}) \cup (X \oplus \overline{\check{X}})$

$\leftrightarrow [(0,0) \notin (\overline{X} \oplus \check{X})] \wedge [(0,0) \notin (X \oplus \overline{\check{X}})]$:

(a) Claim $(0,0) \notin (\overline{X} \oplus \check{X})$:

Assume $(0,0) \in (\overline{X} \oplus \check{X})$

$\rightarrow (0,0) \in \{(a+(-x), b+(-y)) \in W \mid (a,b) \in \overline{X}, (-x,-y) \in \check{X}\}$

$\rightarrow (0,0) \in \{(a-x, b-y) \in W \mid (a,b) \notin X, (x,y) \in X\}$

$\rightarrow \exists (a-x, b-y) = (0,0)$ where $(a,b) \notin X, (x,y) \in X$

$\rightarrow \exists (x,y) = (a,b)$ where $(a,b) \notin X, (x,y) \in X$

which is impossible, since $(x,y) = (a,b) \notin X$ contradicts with $(x,y) = (a,b) \in X$. Therefore, the assumption is wrong, we have that $(0,0) \notin (\overline{X} \oplus \check{X})$.

(b) Claim $(0,0) \notin (X \oplus \overline{\check{X}})$:

Assume $(0,0) \in (X \oplus \overline{\check{X}})$

$\rightarrow (0,0) \in \{(x+(-a), y+(-b)) \in W \mid (x,y) \in X, (-a,-b) \in \overline{\check{X}}\}$

$\rightarrow (0,0) \in \{(x-a, y-b) \in W \mid (x,y) \in X, (-a,-b) \notin \check{X}\}$

$\rightarrow (0,0) \in \{(x-a, y-b) \in W \mid (x,y) \in X, (a,b) \notin X\}$

$\rightarrow \exists (x-a, y-b) = (0,0)$ where $(a,b) \notin X, (x,y) \in X$

$\rightarrow \exists (x,y) = (a,b)$ where $(a,b) \notin X, (x,y) \in X$

which is impossible, since $(x,y) = (a,b) \notin X$ contradicts with $(x,y) = (a,b) \in X$. Therefore, the assumption is wrong, we have that $(0,0) \notin (X \oplus \overline{\check{X}})$.

By (a) and (b), we have $[(0,0) \notin (\overline{X} \oplus \check{X})] \wedge [(0,0) \notin (X \oplus \overline{\check{X}})]$, i.e.

$$I \subset \overline{(\overline{X} \oplus \check{X}) \cup (X \oplus \overline{\check{X}})}.$$

We also know $I \subset I$, then we have

$$I \subset \overline{(\overline{X} \oplus \check{X}) \cup (X \oplus \overline{\check{X}})} \cap I.$$

2. Claim $\overline{(\overline{X} \oplus \dot{X}) \cup (X \oplus \overline{\dot{X}}) \cup \overline{I}} \subset I$:
   Since $I \subset I$, it implies

$$\overline{(\overline{X} \oplus \dot{X}) \cup (X \oplus \overline{\dot{X}}) \cup \overline{I}} = \overline{(\overline{X} \oplus \dot{X}) \cup (X \oplus \overline{\dot{X}})} \cap I \subset I.$$

From (1) and (2), we have

$$I \subset \overline{(\overline{X} \oplus \dot{X}) \cup (X \oplus \overline{\dot{X}})} \cap I$$

and

$$\overline{(\overline{X} \oplus \dot{X}) \cup (X \oplus \overline{\dot{X}}) \cup \overline{I}} = \overline{(\overline{X} \oplus \dot{X}) \cup (X \oplus \overline{\dot{X}})} \cap I \subset I.$$

Thus, by the equivalence of sets, we have $\overline{(\overline{X} \oplus \dot{X}) \cup (X \oplus \overline{\dot{X}})} \cap I = I$.

Case 2: $X \neq \dot{R}$, i.e. $R \neq \dot{X}$.

We want to prove

$$\overline{(\overline{X} \oplus R) \cup (X \oplus \overline{R}) \cup \overline{I}} = \phi \leftrightarrow \overline{(\overline{X} \oplus R) \cup (X \oplus \overline{R})} \cap I = \phi$$

1. Claim $I \not\subset \overline{(\overline{X} \oplus R) \cup (X \oplus \overline{R})}$
   $\leftrightarrow (0,0) \notin \overline{(\overline{X} \oplus R) \cup (X \oplus \overline{R})}$
   $\leftrightarrow (0,0) \in (\overline{X} \oplus R) \cup (X \oplus \overline{R})$
   $\leftrightarrow (0,0) \in (\overline{X} \oplus R) \vee (0,0) \in (X \oplus \overline{R})$:
   Now we assume $(0,0) \notin (\overline{X} \oplus R) \wedge (0,0) \notin (X \oplus \overline{R})$:

   (a) If $(0,0) \notin (\overline{X} \oplus R)$
      $\rightarrow (0,0) \notin \{(a+k, b+l) \mid (a,b) \in \overline{X}, (k,l) \in R\}$
      $\rightarrow (a+k, b+l) \neq (0,0), \forall (a,b) \in \overline{X}, \forall (k,l) \in R$
      $\rightarrow (a,b) \neq (-k,-l), \forall (a,b) \notin X, \forall (k,l) \in R$
      $\rightarrow \forall (k,l) \in R, \exists (a,b) \in X, (a,b) = (-k,-l)$
      $\rightarrow \forall (-k,-l) \in \dot{R}, \exists (a,b) \in X, (a,b) = (-k,-l)$
      $\rightarrow \forall (i,j) \in \dot{R}, \exists (a,b) \in X, (a,b) = (i,j)$
      $\rightarrow ((i,j) \in \dot{R}) \rightarrow (i,j) \in X)$
      $\rightarrow \dot{R} \subset X.$

   (b) If $(0,0) \notin (X \oplus \overline{R})$, then $X \subset \dot{R}$. Since the dilation operation is commutative, by interchanging the variables X and $\dot{R}$ and applying the same procedure as (a), we have $X \subset \dot{R}$.

2. By the above (a) and (b), we have $X = \dot{R}$ which contradicts with $X \neq \dot{R}$. Thus, the assumption is wrong, and we get the following result:
   $\leftrightarrow (0,0) \in (\overline{X} \oplus R) \vee (0,0) \in (X \oplus \overline{R})$
   $\leftrightarrow I \not\subset \overline{(\overline{X} \oplus R) \cup (X \oplus \overline{R})}$
   $\leftrightarrow \overline{(\overline{X} \oplus R) \cup (X \oplus \overline{R})} \cap I = \phi$
   $\leftrightarrow \overline{(\overline{X} \oplus R) \cup (X \oplus \overline{R}) \cup \overline{I}} = \phi.$

Hence, by case 1. and case 2., we have shown that

$$\overline{(\overline{X} \oplus R) \cup (X \oplus \overline{R}) \cup \overline{I}} = \begin{cases} I & if X = \dot{R} \\ \phi & otherwise. \end{cases}$$

Appendix B

Proof of Theorem 2.1: Consider any image transformation (general case):

$$T : \begin{cases} X_1 \longrightarrow A_1 \\ X_2 \longrightarrow A_2 \\ \quad\quad \cdot \\ \quad\quad \cdot \\ \quad\quad \cdot \\ X_l \longrightarrow A_l \end{cases}$$

where $X_i \in P(W), A_i \in P(W), i = 1, 2, ..., l$.

If we choose $R_i = \check{X}_i$, $Q_i = \check{A}_i, i = 1, 2, ..., l$, use Lemma 2.1 and some properties of the dilation (i.e. $I \oplus X = X$ and $\phi \oplus X = \phi$), then we have

$$T(X) = \bigcup_{i=1}^{l} \{\overline{(\overline{X} \oplus R_i) \cup (X \ominus \overline{R_i}) \cup \overline{I}} \oplus Q_i\}.$$

Since some images X_i may map into the null image ϕ for a given image transformation; by Lemma 2.1, we have that

$$T(X) = \bigcup_{i=1}^{k} \{\overline{(\overline{X} \oplus R_i) \cup (X \ominus \overline{R_i}) \cup \overline{I}} \oplus Q_i\}$$

where $k \leq l$, $l = \#(P(W))$ is the cardinality of $P(W)$.

Appendix C

Proof of Theorem 2.2: This can be shown in a very straight forward way. Any image is a set of image points and is the union of point images (consisting one and only one image point). A point image $\{(i, j)\}$ can be written as

$$\{(i, j)\} = A^i B^j.$$

Hence, the union of all point images which are contained in X is the image X. For example, an image $X = \{(2, 0), (1, -1), (-1, 2)\}$ is denoted by

$$X = A^2 \cup AB^{-1} \cup A^{-1}B^2.$$

Appendix D

1. Properties of Complement and Difference

The complement $^-$, a unary operation, is decreasing and shift variant (considering the outside of an image). The difference X/R, a binary operation, is increasing (but decreasing with respect to the reference image R), antiextensive with respect to X, and shift variant (the reference image R is fixed once it is given). Note that the difference operation is not commutative, not associative, and not distributive over other operations. Furthermore, the difference operation is more complicated than the complement. Hence, it is preferable to employ the complement as a fundamental operation, but not the difference. The major properties of the complement and the difference are listed in the following:

1. $\overline{X} = W/X$
2. $X/R = X \cap \overline{R}$
3. $\overline{\phi} = W$
4. $\overline{W} = \phi$
5. $\overline{\overline{X}} = X$ (idempotent for twice complements)
6. $X/\phi = X$ (idempotent for a given reference image $R = \phi$)
7. $X/X = \phi$
8. $X \subset Y \rightarrow \overline{Y} \subset \overline{X}$ (decreasing)
9. $X \subset Y \rightarrow X/R \subset Y/R$ (increasing)
10. $X/R \subset X$ (antiextensive)
11. $X \subset R \rightarrow X/R = \phi$
12. $\overline{X \cap R} = \overline{X} \cup \overline{R}$
13. $\overline{X \cup R} = \overline{X} \cap \overline{R}$
14. $X \cap \overline{X} = \phi$
15. $X \cup \overline{X} = W$
16. $\overline{X \oplus R} = \overline{X} \ominus \dot{R}$ where $\dot{R} = \{(-x, -y) \mid (x, y) \in R\}$
17. $\overline{X \oplus R} = \overline{X} \ominus \dot{R}$ where $\dot{R} = \{(-x, -y) \mid (x, y) \in R\}$

2. Properties of Union and Intersection

The union $\cup$, a binary operation, is increasing, extensive, shift variant, idempotent, commutative, associative, and distributive over intersection. The intersection $\cap$, a binary operation, is increasing, antiextensive, shift variant, idempotent, commutative, associative, and distibutive over union. The major properties of the union and the intersection are listed in the following:

1. $X \cup \phi = X$
   $X \cap \phi = \phi$
2. $X \cup X = X$
   $X \cap X = X$
3. $X \cup R = R \cup X$ (commutative)
   $X \cap R = R \cap X$ (commutative)
4. $X \cup (R \cup Q) = (X \cup R) \cup Q$ (associative)
   $X \cap (R \cap Q) = (X \cap R) \cap Q$ (associative)
5. $X \cup W = W$
   $X \cap W = X$ (idempotent for a given reference image $R = W$)

6. $X \cup (R \cap Q) = (X \cup R) \cap (X \cup Q)$ (distributive)
   $X \cap (R \cup Q) = (X \cap R) \cup (X \cap Q)$ (distributive)

7. $X \subset X \cup R$ (extensive)
   $X \cap R \subset X$ (antiextensive)

8. $X \subset Y \rightarrow X \cup R \subset Y \cup R$ (increasing)
   $X \subset Y \rightarrow X \cap R \subset Y \cap R$ (increasing)

9. $X \subset R \rightarrow X \cup R = R$
   $X \subset R \rightarrow X \cap R = X$

10. $R \subset X \wedge Q \subset X \rightarrow R \cup Q \subset X$
    $X \subset R \wedge X \subset Q \rightarrow X \subset R \cup Q$

11. $R \subset X \wedge Q \subset Y \rightarrow R \cup Q \subset X \cup Y$
    $R \subset X \wedge Q \subset Y \rightarrow R \cap Q \subset X \cap Y$

3. Properties of Dilation and Erosion

The dilation $\oplus$, a binary operation, is increasing, extensive for a given reference image R which contains the elementary image I, shift invariant, commutative, associative, distributive over union, and possesses an identity which is I. The erosion $\ominus$, a binary operation, is shift invariant, increasing (but decreasing with respect to the refernce image R), antiextensive for a given reference image R which contains the elementary image I. But, in general, the erosion is not commutative, not associative, not distibutive over other operations, and does not possesses a left identity. The major properties of the union and the intersection are listed in the following:

1. $X \oplus R = R \oplus X$ (commutative)
   $X \ominus R \neq R \ominus X$ (in general)

2. $(X \oplus R) \oplus Q = X \oplus (R \oplus Q)$ (associative)
   $(X \ominus R) \ominus Q \neq X \ominus (R \ominus Q)$ (in general)
   $(X \ominus R) \ominus Q = (X \ominus Q) \ominus R$

3. $X \oplus (R \cup Q) = (X \oplus R) \cup (X \oplus Q)$ (distributive)
   $X \ominus (R \cup Q) = (X \ominus R) \cap (X \ominus Q)$
   $X \ominus (R \oplus Q) = (X \ominus R) \ominus Q$

4. $X \oplus I = X = I \oplus X$ (identity)
   $X \ominus I = X \neq I \ominus X$ (in general)

5. $X \oplus \phi = \phi = \phi \oplus X$
   $X \ominus \phi = W \neq \phi \ominus X$ (in general)

6. $X \subset X \oplus R$ when $I \subset R$ (extensive)
   $X \ominus R \subset X$ when $I \subset R$ (antiextensive)

7. $X \subset Y \rightarrow X \oplus R \subset Y \oplus R$ (increasing)
   $X \subset Y \rightarrow X \ominus R \subset Y \ominus R$ (increasing)

8. $R \subset Q \rightarrow X \oplus R \subset X \oplus Q$
   $R \subset Q \rightarrow X \ominus Q \subset X \ominus R$

9. $X \oplus (R \cap Q) \subset (X \oplus R) \cap (X \oplus Q)$ (distributive inequality)
   $X \ominus (R \cap Q) \supset (X \ominus R) \cup (X \ominus Q)$
   $(X \cup Y) \ominus R \supset (X \ominus R) \cap (Y \ominus R)$
   $(X \ominus R) \oplus Q \subset (X \oplus R) \ominus Q$
   Remark: "$\supset$" means "contains".

4. Properties of some standard operations

1. The symmetric difference is shift variant (with a fixed reference image R), commutative and associative. Symbolically,

   (a) $X \triangle R = R \triangle X$
   (b) $X \triangle (R \triangle Q) = (X \triangle R) \triangle Q$
   (c) $X \triangle \phi = X$
   (d) $X \triangle X = \phi$
   (e) $X \triangle \overline{X} = W$
   (f) $X \triangle W = \overline{X}$
   (g) $X \cap (R \triangle Q) = (X \cap R) \triangle (X \cap Q)$
   (h) $X \cup (R \triangle Q) \neq (X \cup R) \triangle (X \cup Q)$ (in general)
   (i) $X \triangle R = Y \triangle R \rightarrow X = Y$

2. The opening $\circ$ is shift invariant, increasing, anti-extensive and idempotent. The closing $\bullet$ is shift invarinat, extensive, and idempotent. Symbolically,

   (a) $X \circ R \subset X \subset X \bullet R$
   (b) $X \subset Y \rightarrow X \circ R \subset Y \circ R$
   (c) $X \subset Y \rightarrow X \bullet R \subset Y \bullet R$
   (d) $(X \circ R) \circ R = X \circ R$
   (e) $(X \bullet R) \bullet R = X \bullet R$

3. The thinning is shift invariant and antiextensive. The thickening is shift invariant and extensive. The major properties are in the following:

   (a) $X \circledcirc R \subset X \subset X \odot R$
   (b) $X \subset Y \rightarrow X \circledcirc R \subset Y \circledcirc R$
   (c) $X \subset Y \rightarrow X \odot R \subset Y \odot R$
   (d) If $R \subset Q$ (which means $R_1 \subset Q_1$ and $R_2 \subset Q_2$), then we have:
   $R \subset Q \rightarrow X \circledcirc R \subset X \circledcirc Q \subset X \subset X \odot Q \subset X \odot R$.
   (e) $\overline{X \odot R} = \overline{X} \circledcirc R^*$ where $R = \{R_1, R_2\}$ and $R^* = \{R_2, R_1\}$.

Appendix E

Proof of Theorem 3.3: We can easily see that (2) in Therorem 3.3 is a generalization of (1) in Theorem 3.3. (1) is used for exactly matching shapes (or templates) with shift invariance; (2) is generalized to more general cases. For example, to consider noise and to have rotational invariance, we can choose the family $\{R(\theta)\}$ to incorporate all aspect reference image pairs. In the following, we prove (1) and then (2) will follow from it directly. The proof will demonstrate the mathematical correspondance between boolean logic and BIA. The notations $x(i,j)$ and $r(i,j)$ will be used to represent the binary values (0 or 1) of pixels at coordinate (i,j) of image functions which correspond to the images X and R in BIA notations.

First, let us use the boolean logic XOR (excusive or) operation, i.e.

$$x(i,j) \text{ XOR } r(i,j) = (\bar{x}(i,j) \wedge r(i,j)) \vee (x(i,j) \wedge \bar{r}(i,j)),$$

to achieve the pixel-wise comparison where the ouput value with '0' means that '$x(i,j)$' matches '$r(i,j)$' and the output value with '1' means that '$x(i,j)$' does not match '$r(i,j)$'.

Second, to check the occurence of the shape (defined by R with M) in the tested image X at coordinate (i,j), we have to shift the origin of the shape to the coordinate (i,j) in X. Then the process of the comparison of the shape and the subimage in X (limited in the mask M) and the indication of "match" (0) and "not match" (1) will be performed by

$$\bigvee_{(k,l)\in M} (\bar{x}(i+k,j+l) \wedge r(k,l)) \vee \bigvee_{(k,l)\in M} (x(i+k,j+l) \wedge \bar{r}(k,l)).$$

If the above equation is considered as a binary operation operating on two images $x(i,j)$ and $r(i,j)$, then this operation is not commutative; in order to achieve the commutativity, we change (k,l) with $(-k,-l)$ and denote $\check{r}(k,l) = r(-k,-l)$:

$$\bigvee_{(-k,-l)\in \check{M}} (\bar{x}(i-k,j-l) \wedge \check{r}(k,l)) \vee \bigvee_{(-k,-l)\in \check{M}} (x(i-k,j-l) \wedge \bar{\check{r}}(k,l)).$$

If the output value of the above equation is '0', then it means that the location (i,j) of the image X has the occurrence of the shape (defined by R and M); if '1', the shape is not occurred at (i,j).

Third, let us run over all coordinates (i,j) (i.e. for all $(i,j) \in W$ the universal image) and then the union of those coordinates with value '0' would be the answer. The value '0' at a coordinate (i,j) corresponds to the null image in set notation and the value '1' at a coordinate (i,j) corresponds to the point image $\{(i,j)\}$. For convenience, in the following we mix the notations of boolean logic functions and set notations; if the output of a boolean logic expression is '0', it represents the null image ϕ; if '1', it represents the point image $\{(i,j)\}$. Thus, we have

$$\bigcup_{(i,j)\in W} \Big(\bigvee_{(-k,-l)\in \check{M}} (\bar{x}(i-k,j-l) \wedge \check{r}(k,l)) \vee \bigvee_{(-k,-l)\in \check{M}} (x(i-k,j-l) \wedge \bar{\check{r}}(k,l))\Big)$$

which is the same as

$$\bigcup_{(i,j)\in W} \Big(\bigvee_{(-k,-l)\in \check{M}} (\bar{x}(i-k,j-l) \wedge \check{r}(k,l))\Big) \cup \bigcup_{(i,j)\in W} \Big(\bigvee_{(-k,-l)\in \check{M}} (x(i-k,j-l) \wedge \bar{\check{r}}(k,l))\Big).$$

Since $x(i,j) \neq 0$ only when $(i,j) \in X$ and $\dot{r}(k,l) \neq 0$ only when $(k,l) \in \dot{R}$, we have

$$\begin{aligned}
\bigcup_{(i,j)\in W} \Big(\bigvee_{(-k,-l)\in \dot{M}} (\overline{x}(i-k,j-l) \wedge \dot{r}(k,l))\Big) &= \{(i,j) \mid (i-k,j-l) \in \overline{X}, (k,l) \in \dot{R}\} \\
&= \{(i+k,j+l) \mid (i,j) \in \overline{X}, (k,l) \in \dot{R}\} \\
&= \overline{X} \oplus \dot{R}.
\end{aligned}$$

Similarly, we have

$$\bigcup_{(i,j)\in W} \Big(\bigvee_{(-k,-l)\in \dot{M}} (x(i-k,j-l) \wedge \dot{r}(k,l)\Big) = X \oplus (\dot{M}/\dot{R}).$$

Hence, if we use '0' to indicate "match", we have

$$(\overline{X} \oplus \dot{R}) \cup (X \oplus (\dot{M}/\dot{R}));$$

if we use '1' to indicate "match",then we have

$$\overline{(\overline{X} \oplus \dot{R}) \cup (X \oplus (\dot{M}/\dot{R}))}.$$

Thus, the locations of a shape, which is defined by a non-null reference image R with a non-null reference image (called mask) M and $R \subset M \subset W$, are the image points in the following

$$\overline{(\overline{X} \oplus \dot{R}) \cup (X \oplus (\dot{M}/\dot{R}))} = \overline{(\overline{X} \oplus \dot{R}) \cup (X \oplus \overline{\overline{\dot{M}} \cup \dot{R}})} = (X \ominus R) \cap (\overline{X} \ominus (M/R)).$$

A more intuitive illustration is that the foreground X should match R by $X \ominus R$ (using multiple-input AND gates to examine the locations where the 1's should be), while the background $\overline{X}$ should match M/R by $\overline{X} \ominus (M/R)$. Combining both results by the intersection (AND), we then implement the shape recognition by $(X \ominus R) \cap (\overline{X} \ominus (M/R))$. Replacing R by R_1 and (M/R) by R_2, we obtain the hit or miss transform (template matching) for shape recognition.

Appendix F

Proof of Theorem 3.4:

(1) The straight forward way for removing the "pepper" noise is the thinning operation $X \circledcirc R_4$ (or $X \circledcirc R_8$). Follow this, we have

$$\begin{aligned}
X \circledcirc R_4 &= \overline{\overline{X} \cup \overline{(\overline{X} \oplus I) \cup (X \oplus M_4)}} \\
&= \overline{\overline{X} \cup \overline{\overline{X} \cup (X \oplus M_4)}} \\
&= \overline{\overline{X} \cup (X \cap \overline{\overline{X} \oplus M_4})} \\
&= \overline{(\overline{X} \cup X) \cap (\overline{X} \cup \overline{\overline{X} \oplus M_4})} \\
&= \overline{W \cap (\overline{X} \cup \overline{\overline{X} \oplus M_4})} \\
&= \overline{\overline{X} \cup \overline{\overline{X} \oplus M_4}}.
\end{aligned}$$

(2) The straight forward way for removing the "pepper" noise is the thickening operation $X \odot Q_4$ (or $X \odot Q_8$). Follow this, we have

$$\begin{aligned} X \odot Q_4 &= X \cup \overline{(\overline{X} \oplus M_4) \cup (X \oplus I)} \\ &= X \cup \overline{(\overline{X} \oplus M_4) \cup X} \\ &= X \cup (\overline{\overline{X} \oplus M_4} \cap \overline{X}) \\ &= (X \cup \overline{\overline{X} \oplus M_4}) \cap (X \cup \overline{X}) \\ &= (X \cup \overline{\overline{X} \oplus M_4}) \cap W \\ &= (X \cup \overline{\overline{X} \oplus M_4}). \end{aligned}$$

(3)The straight forward way for removing the "salt and pepper" noise the difference of $X \odot Q_4$ by $X \circledast R_4$ (or the difference of $X \odot Q_8$ by $X \circledast R_8$). By a similar procedure as above we can achieve the desired result.

Appendix G

Proof of Theorem 3.5: To extract the region whose sizes are between two reference images R and Q, the straight forward way is to design a morphological band pass filter (Theorem 3.2):

$$(X \circ R)/(X \circ Q) = ((X \ominus R) \oplus R)/((X \ominus Q) \oplus Q).$$

To obtain the locations of those desired regions, we then perform the skelotonization:

$$S(((X \ominus R) \oplus R)/((X \ominus Q) \oplus Q)) = S((X \ominus R)/((X \ominus Q) \oplus Q))) = S(\overline{(\overline{X} \oplus \dot{R}) \cup (\overline{\overline{X} \oplus \dot{Q}} \oplus Q)}).$$

References

[1] K. S. Huang, B. K. Jenkins, A. A. Sawchuk, "Optical Cellular Logic Architectures Based on Binary Image Algebra ", *Proc. IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Machine Intelligence*, Seattle, October, pp. 19-26, 1987.

[2] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, Inc., New York, 1982.

[3] R. M. Lougheed, D. L. McCubbrey, and S. R. Sternberg, " Cytocomputers: Architectures for Parallel Image Processing," *Proc. IEEE Workshop Picture Data Description and Management*, CA, pp. 281-286, 1980.

[4] G. Matheron, *Random Sets and Integral Geometry*, Wiley, New York, 1975.

[5] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image Analysis Using Mathematical Morphology," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 4, pp. 532-550, 1987.

[6] K. Preston and M. J. B. Duff, *Modern Cellular Automata — Theory and Applications*, Plenum Press, New York, 1984.

[7] A. Burks ed., *Essays on Cellular Automata*, Univ. of Illinois Press, 1970.

[8] G. X. Ritter and P. D. Gader, "Image Algebra Techniques for Parallel Image Processing",*Journal of Parallel and Distributed Computing*, Vol. 4, No. 1, pp. 7-44, 1987.

[9] C. R. Giardina, "The Universal Imaging Algebra", *Pattern Recognition Letters*, Vol. 2, pp. 165-172, 1984.

[10] T. Agui, et al., "An Algebraic Approach to the Generation and Description of Binary Pictures", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. PAMI-4, No. 6, pp. 635-641, 1982.

[9] C. R. Giardina, "The Universal Imaging Algebra", *Pattern Recognition Letters*, Vol. 2, pp. 165-172, 1984.

[10] T. Agui, et al., "An Algebraic Approach to the Generation and Description of Binary Pictures", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. PAMI-4, No. 6, pp. 635-641, 1982.

[11] S. Sternberg, "Biomedical Image Processing," *IEEE Computer*, Jan. 1982, pp. 22-34.

[12] S. Sternberg, "An Overview of Image Algebra and Related Architectures", in *Integrated Technology for Parallel Image Processing*, Edited by S. Levialdi, Academic Press, New York, 1985, pp.79-100.

[13] J. Klein and J. Serra, "The texture analyzer," *J. Microscopy*, Vol. 95, No. 2, 1972, pp.349-356.

[14] J. Mandeville, "Novel Method for Automated Optical Inspection of Printed Circuits," *IBM Research Report RC-9900*, IBM T. J. Waston Research Lab., Yorktown Heights, N. Y., 1983.

[15] "Morphological Systems Software," A Session in *Computer Architecture for Pattern Analysis and Image Database Management*, 1985 IEEE Computer Society Workshop, pp. 429-468.

[16] "Morphological Systems Hardware," A Session in *Computer Architecture for Pattern Analysis and Image Database Management*, 1985 IEEE Computer Society Workshop, pp. 469-500.

[17] P. W. Verbeek, "Implementation of Cellular-Logic Operators Using 3*3 Convolution and Table Lookup Hardware," *Computer Vision, Graphics, and Image Processing*, Vol. 27, 1984, pp. 115-123.

[18] S. R. Sternberg and J. Serra ed., "Special Section on Mathematical Morphology", *Computer Vision, Graphics, and Image Processing*, Vol. 35, 1986, pp. 273-383.

[19] K. S. Huang, B. K. Jenkins, A. A. Sawchuk, "An Image Algebra Representation of Parallel Optical Binary Arithmetic", submitted to *Applied Optics*.

[20] J. von Neumann, "The General Logical Theory of Automata," in *Cerebral Mechanisms in Behavior-The Hixon Symposium* , L. A. Jeffress, Ed. New York: Wiley, 1951.

[21] J. von Neumann, *Theory of Self-reproducing Automata*, edited by A. W. Burks, Univ. of Illinois Press, 1966.

[22] S. H. Unger, "A Computer Oriented Toward Spatial Problems," *Proc. IRE*, Vol.46,1958, pp. 1744-1750.

[23] K. Preston Jr. et al., "Basics of Cellular Logic with Some Application's in Medical Image Processing," *Proc. of IEEE*, Vol. 67, No. 5, 1979, pp. 826-856.

[24] A. Rosenfeld, "Parallel Image Processing Using Cellular Arrays," *IEEE Computer*, Jan. 1983, pp. 14-20.

[25] K. Preston Jr., "Cellular Logic Computers for Pattern Recognition," *IEEE Computer*, Jan. 1983, pp.36-47.

[26] B. K. Jenkins and A. A. Sawchuk, "Optical Cellular Logic Architectures for Image Processing," *IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, Florida, , pp. 61-65, Nov. 1985.

[27] A. A. Sawchuk and B. K. Jenkins, "Optical Cellular Logic Processors," *Optical Society of America 1985 Annual Meeting*, Washington, D.C..

[28] A. A. Sawchuk and T. C. Strand, "Digital Optical Computing," *Proc. IEEE*, Vol. 72, pp. 758-779,1984.

[32] G. Gilbert, *Mordern Algebra with Applications*, John Wiley & Sons, New York, 1976.

[33] A. Rosenfeld, "Connectivity in Digital Pictures," *J. Assoc. Comput. Mach.*, Vol. 17, pp. 146-160, 1970.

[34] A. Rosenfeld, A. C. Kak, *Digital Picture Processing*, Academic Press, 1982.

[35] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons. 1973.

[36] S. Levialdi, "On shrinking of binary patterns," *Commun. ACM*, Vol. 15, pp. 7-10, 1972.

[37] A. Huang, "Parallel Algorithms for Optical Digital Computers," in *Technical Digest. IEEE Tenth International Optical Computing Conference*, pp. 13-17, 1983.

[38] K.-H. Brenner, A. Huang, and N. Streibl, "Digital optical computing with symbolic substitution," *Applied Optics*, Vol. 25, pp. 3054-3060, 1986.

[39] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, 1975.

[40] W. K. Pratt, *Digital Image Processing*, John Wiley & Sons, 1978.

[41] J. W. Goodman, *Introduction to Fourier Optics*, McGraw-Hill, 1968.

[42] K. S. Huang, B. K. Jenkins, A. A. Sawchuk, "A Cellular Hypercube Architecture for Image Processing", *Proc. Soc. Photo-Opt. Instr. Eng.*, Vol. 829, August, 1987.

[43] B. K. Jenkins, et al., "Sequential Optical Logic Implementation," *Applied Optics*, Vol. 23, No. 19, pp. 3455-3464, 1984.

[44] T. E. Bell, "Optical Computing: A Field in Flux," *IEEE Spectrum*, Vol. 23, No. 8, pp. 34-57, 1986.

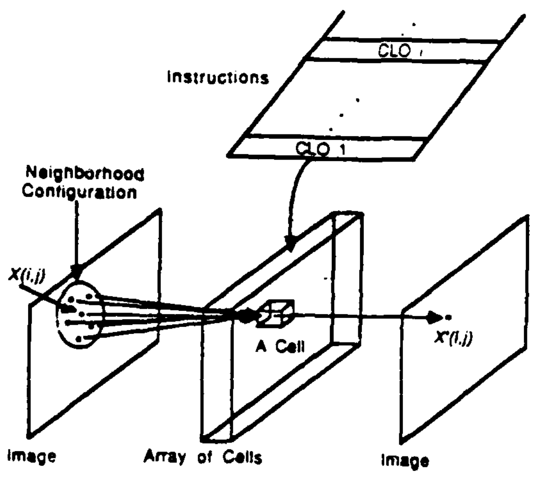


Figure 1: A sequential process of cellular logic operations (CLO). The value $X'(i,j)$ is determined by the corresponding $X(i,j)$ in the original image along with the values of its neighbors.

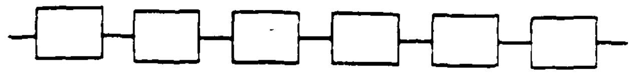
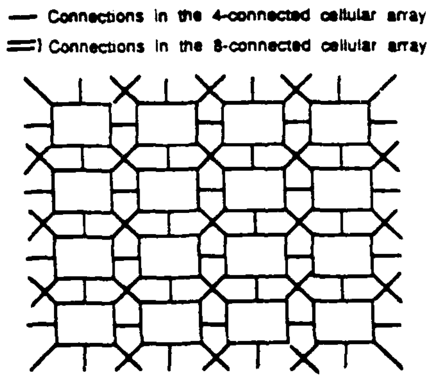


Figure 2(a): A cellular string. It requires only a 1-D interconnection mechanism. Each cell connects with its two nearest cells.



Connections in the 4-connected cellular array
Connections in the 8-connected cellular array

Figure 2(b): A cellular array. It requires a 2-D interconnection mechanism. Each cell connects with its 4 or 8 nearest cells.

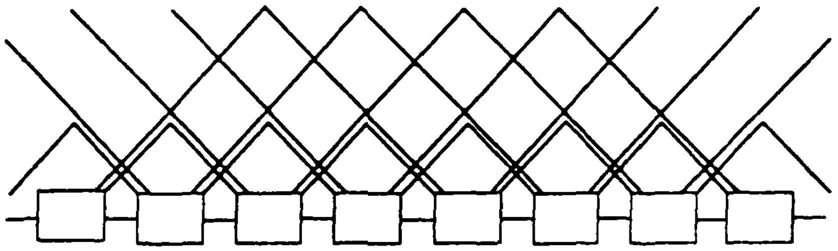


Figure 2(c): A one-dimensional cellular hypercube [24]. Each cell connects with cells at distances $1, 2, 4, 8, \ldots 2^k$ from it. Here, only the connections with distances 1, 2, and 4 are shown.

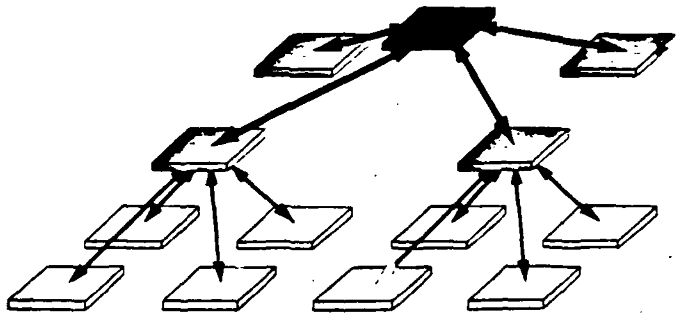


Figure 2(d): A two-dimensional cellular pyramid. It consists of stages of arrays with connections between two adjacent stages and requires a 3-D interconnection mechanism.

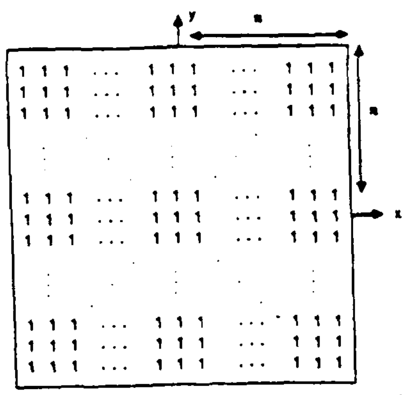


Figure 3: The universal image W. It has $(2n+1) \times (2n+1)$ image points and n is a positive integer.

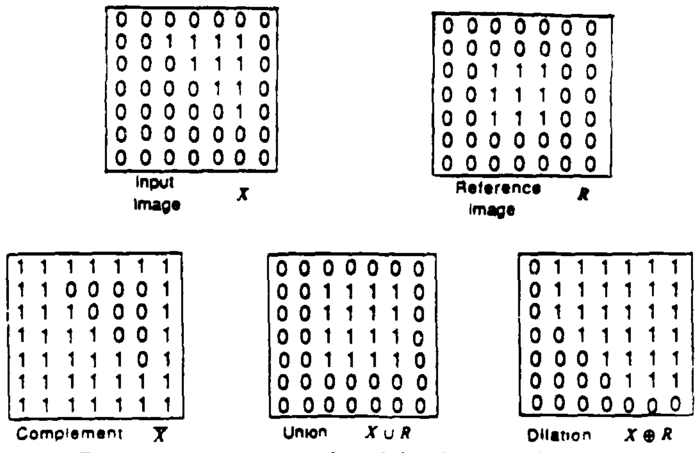
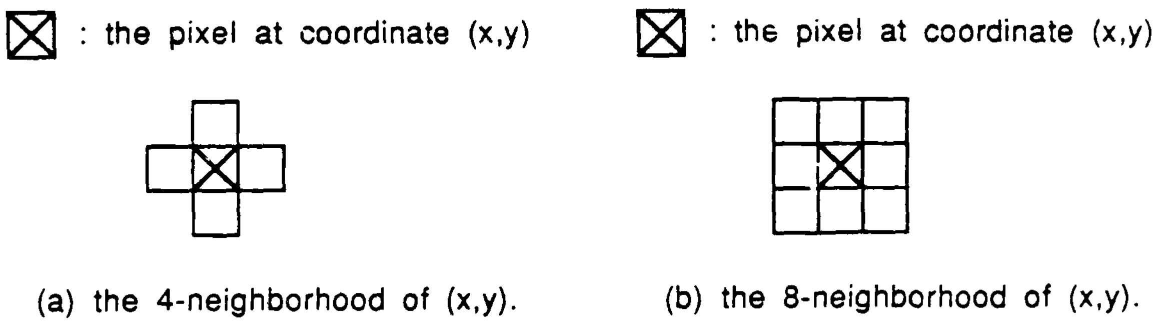


Figure 4: An example of fundamental operations: complement $^{-}$, union $\cup$, and dilation $\oplus$.


: the pixel at coordinate (x,y)
: the pixel at coordinate (x,y)
(a) the 4-neighborhood of (x,y).
(b) the 8-neighborhood of (x,y).
Figure 5: The 4-neighborhood and 8-neighborhood of an image point (x, y).

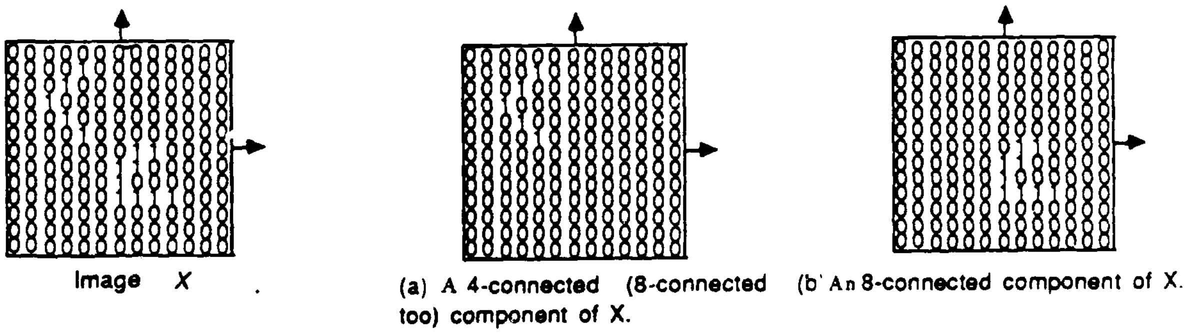


Image X
(a) A 4-connected (8-connected too) component of X.
(b) An 8-connected component of X.
Figure 6: The 4-connected component and 8-connected component of an image.

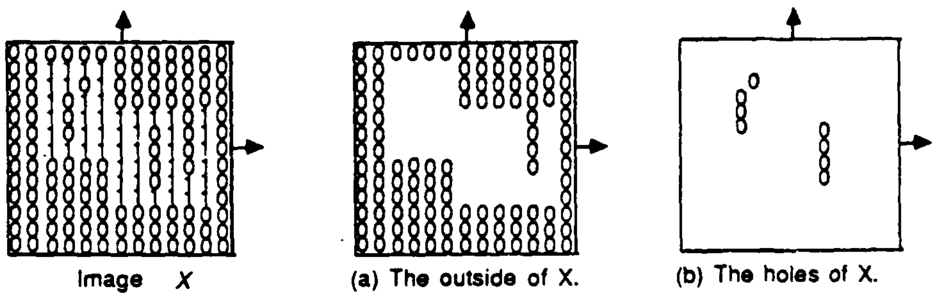


Figure 7: The outside and holes of an image.

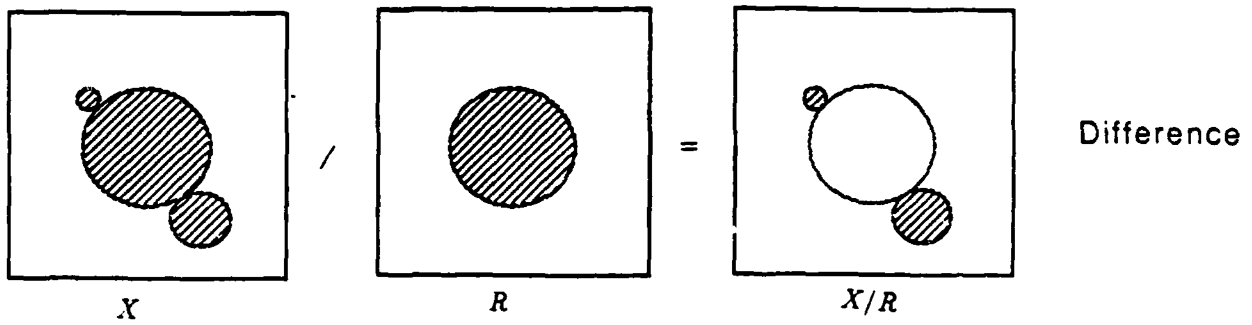


Figure 8(a): Difference.

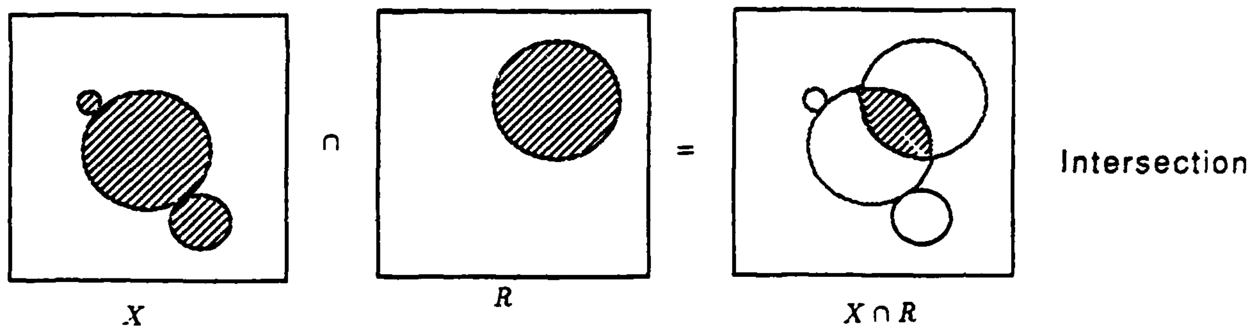


Figure 8(b): Intersection.

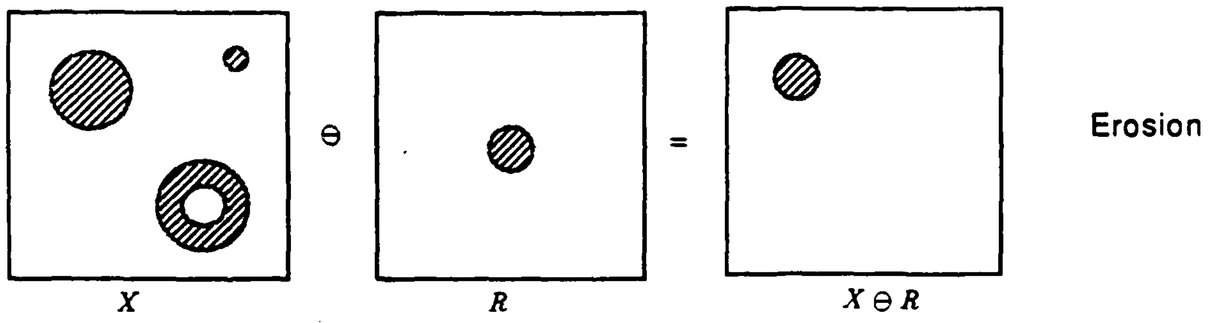


Figure 8(c): Erosion.

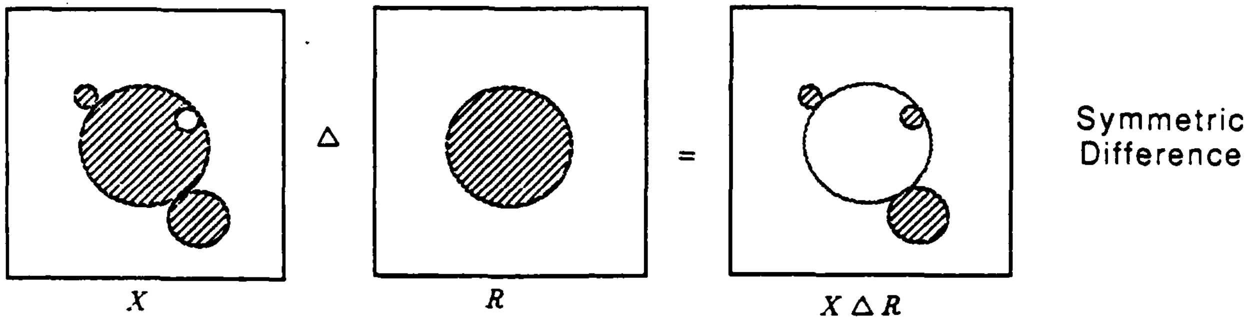


Figure 8(d): Symmetric difference.

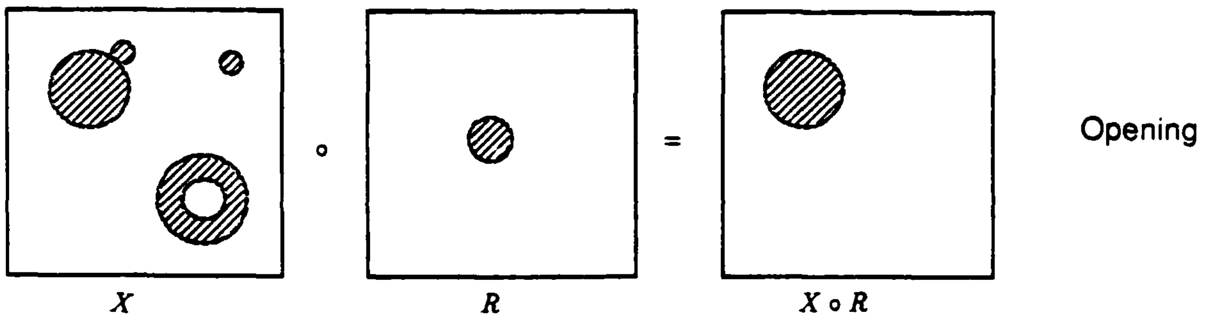


Figure 8(e): Opening.

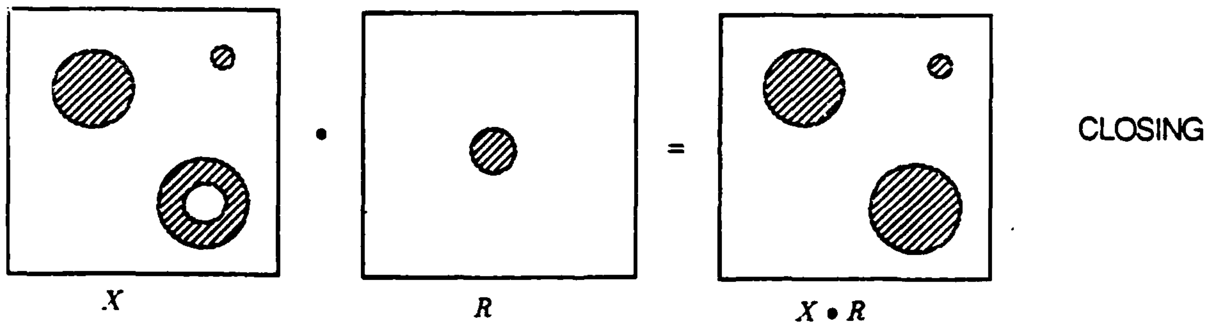


Figure 8(f): Closing.

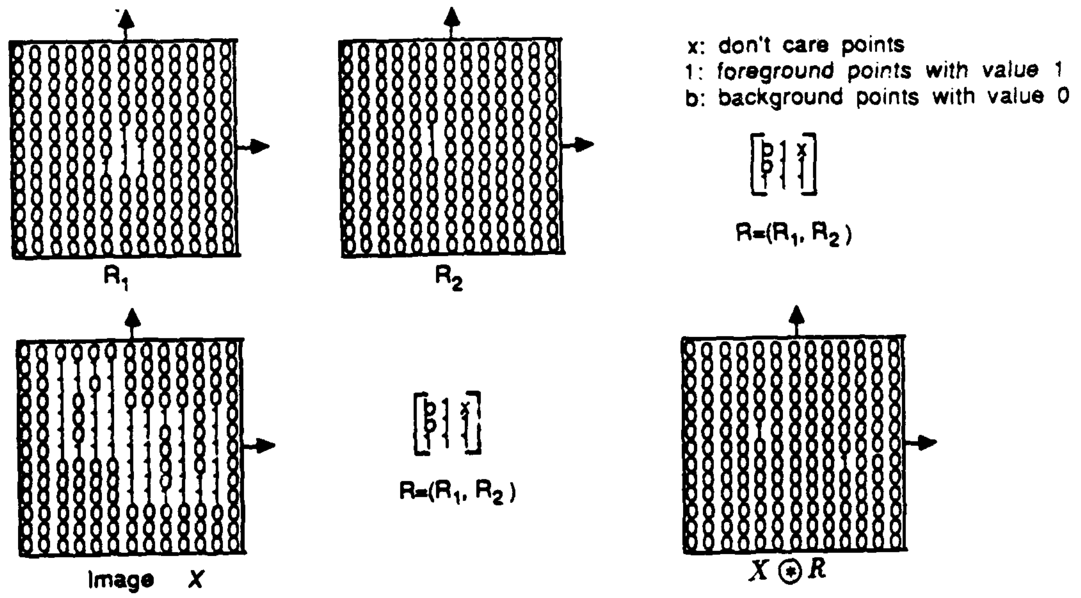

R₁
R₂
x: don't care points
1: foreground points with value 1
b: background points with value 0
R=(R₁, R₂)
Image X
R=(R₁, R₂)
X ⊛ R

Figure 8(g): Hit or miss transform (template matching).

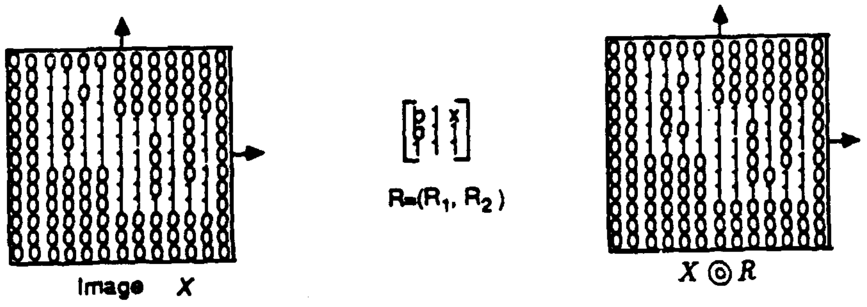


Figure 8(h): Thinning.

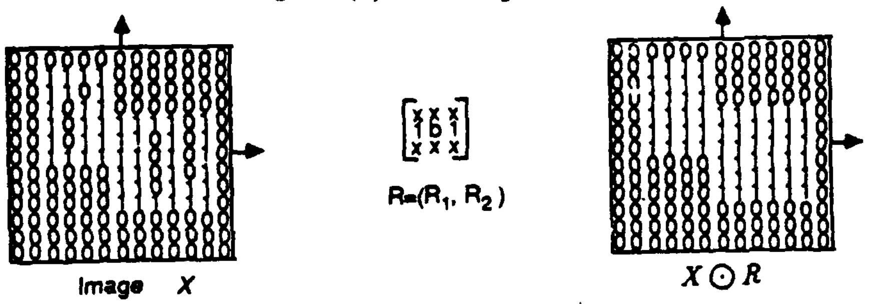


Figure 8(i): Thickening.

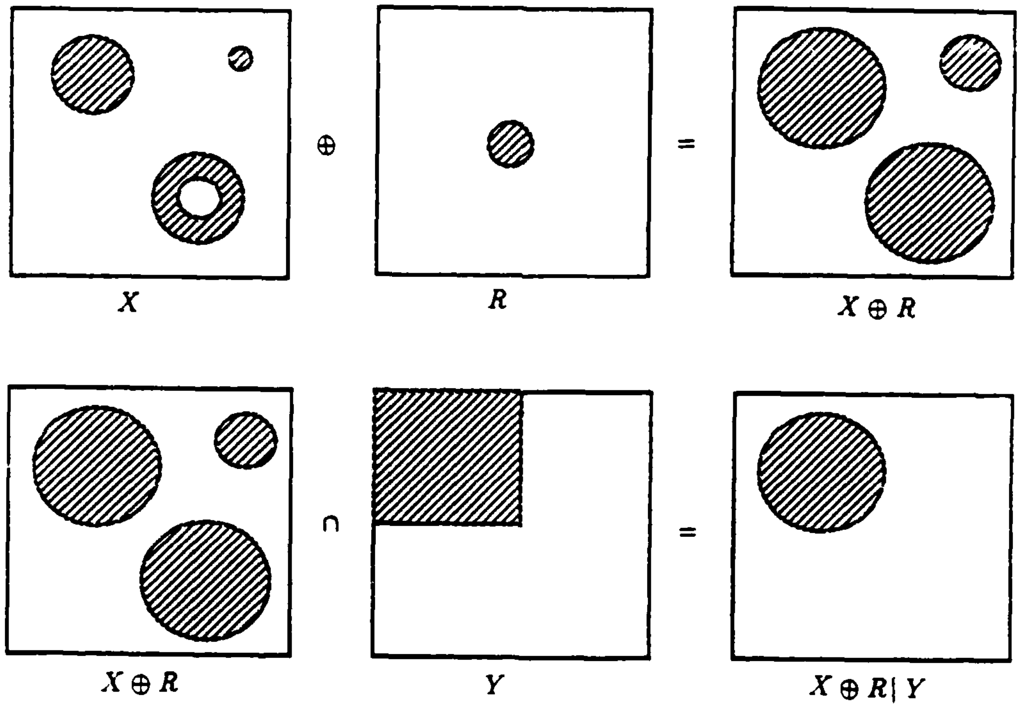


Figure 8(k): A conditional dilation.

Operations / Properties	Complement $\overline{X}$	Union $X \cup R$	Dilation $X \oplus R$	Difference X/R	Intersection $X \cap R$	Erosion $X \ominus R$
Increasing	No	Yes	Yes	Yes	Yes	Yes
Decreasing	Yes	No	No	No	No	No
Extensive	No	Yes	Yes (if $R \supset I$)	No	No	No
Antiextensive	No	No	No	Yes	Yes	Yes (if $R \supset I$)
Idempotent	No	Yes	No	Yes	Yes	No
Shift invariant	No	No	Yes	No	No	Yes
Homotopic	No	No	No	No	No	No
Commutative	No	Yes	Yes	No	Yes	No
Associative	No	Yes	Yes	No	Yes	No
Distributive (with some oper.)	No	Yes (with $\cap$)	Yes (with $\cup$)	No	Yes (with $\cup$, $\triangle$)	No

Table 1(a): Basic properties of the three fundamental operations and of three derived operations (alternative fundamental operations).

Operations / Properties	Symmetric Difference $X \triangle R$	Opening $X \circ R$	Closing $X \bullet R$	Thinning $X \circledcirc R$	Thickening $X \odot R$	Homotopic skeletonization $X \circledcirc (R_\theta)$
Increasing	No	Yes	Yes	Yes	Yes	No
Decreasing	No	No	No	No	No	No
Extensive	No	No	Yes	No	Yes	No
Antiextensive	No	Yes	No	Yes	No	Yes
Idempotent	No	Yes	Yes	No	No	Yes
Shift invariant	No	Yes	Yes	Yes	Yes	Yes
Homotopic	No	No	No	No	No	Yes
Commutative	Yes	No	No	No	No	No
Associative	Yes	No	No	No	No	No
Distributive (with some oper.)	No	No	No	No	No	No

Table 1(b): Basic properties of some standard derived operations.

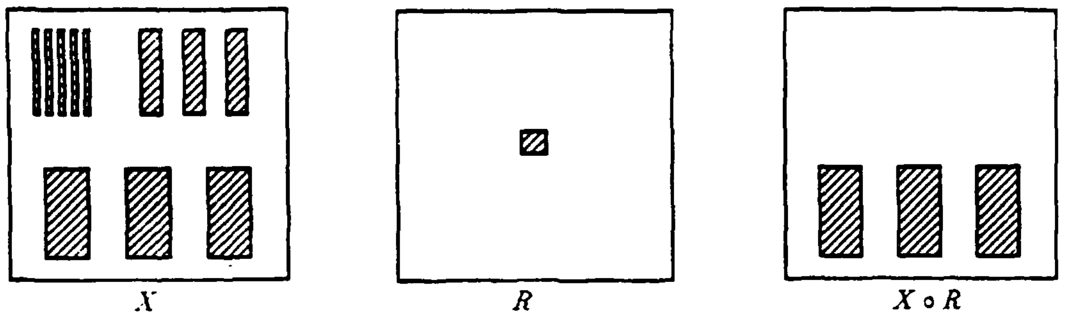


Figure 9(a): One kind of morphological low pass filter (opening).

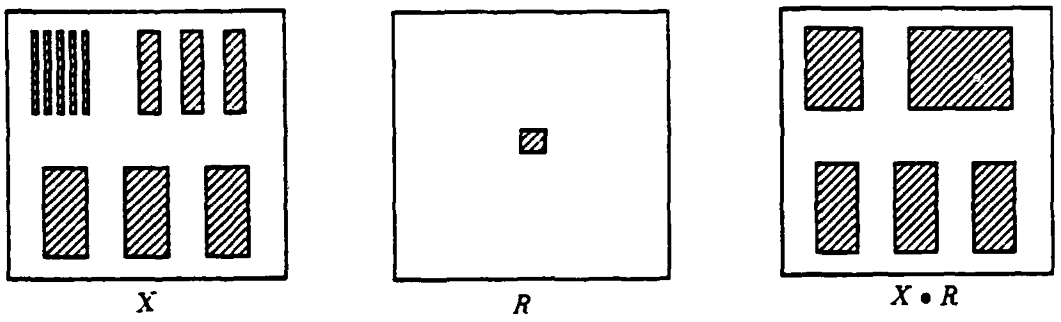


Figure 9(b): A second kind of morphological low pass filter (closing).

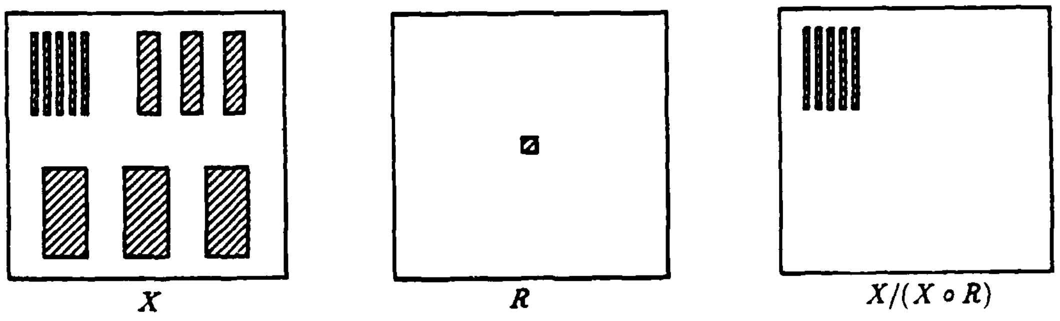


Figure 9(c): A morphological high pass filter.

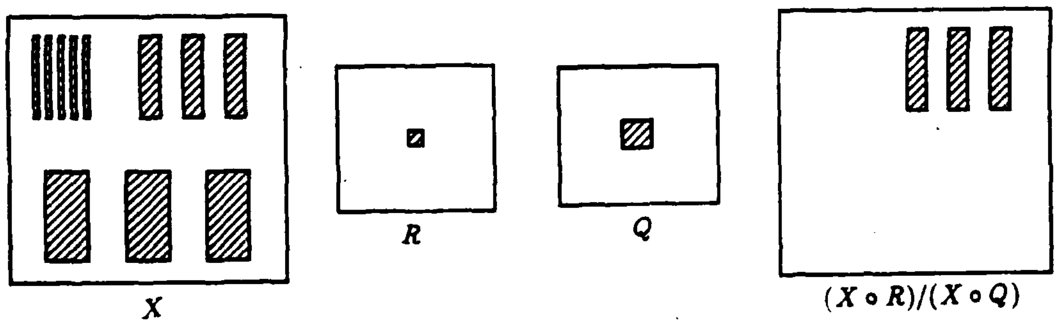
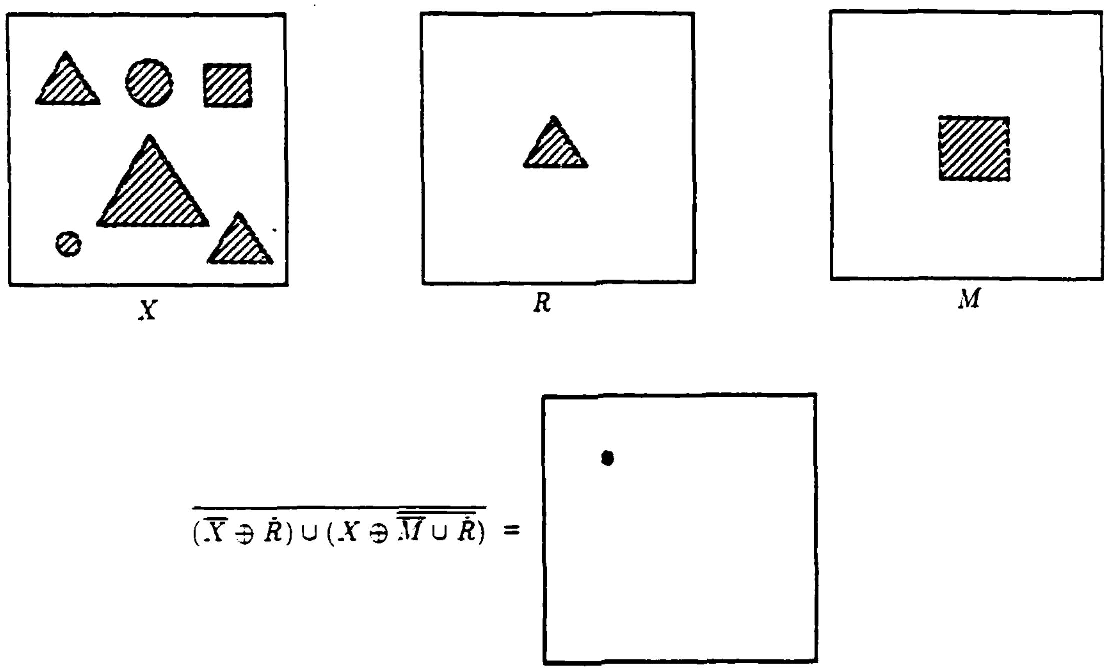


Figure 9(d): A morphological band pass filter.



$$\overline{(\overline{X} \ni \dot{R}) \cup (X \ni \overline{\overline{M} \cup \dot{R}})} =$$

Figure 10(a): One kind of shape recognition. R represents the shape to be identified, and must lie entirely and exclusively in the mask defined by M.

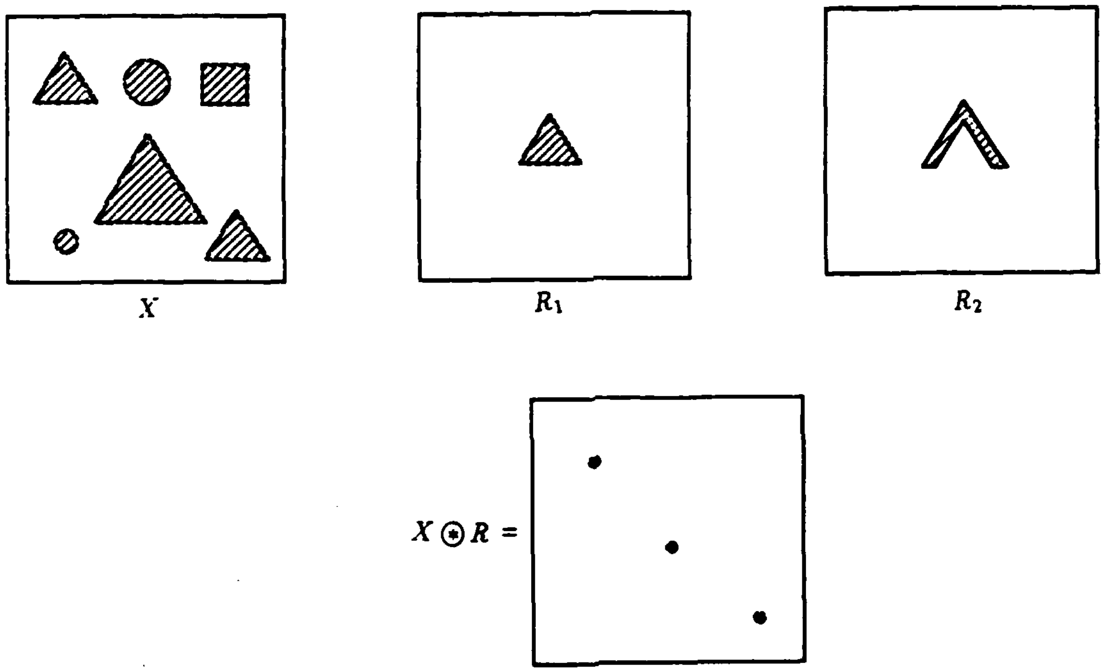


Figure 10(b): Hit or miss transform, which recognizes locations of foreground points given by R_1 in conjunction with background points given by R_2.

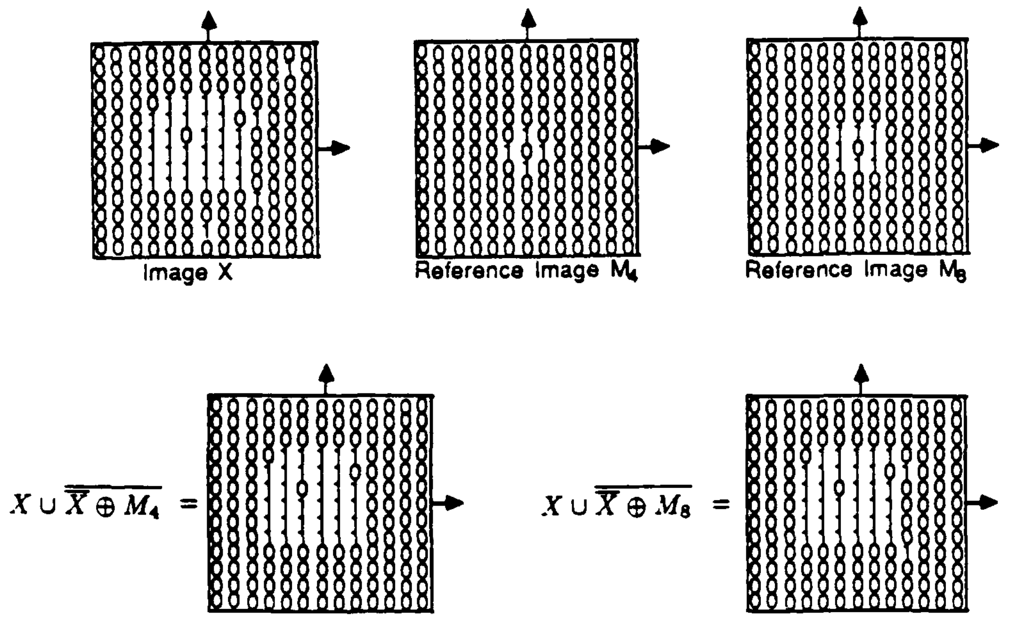


Image X
Reference Image M4
Reference Image M8
$X \cup \overline{\overline{X} \oplus M_4}$ =
$X \cup \overline{\overline{X} \oplus M_8}$ =

Figure 11(a): "Salt" noise removal.

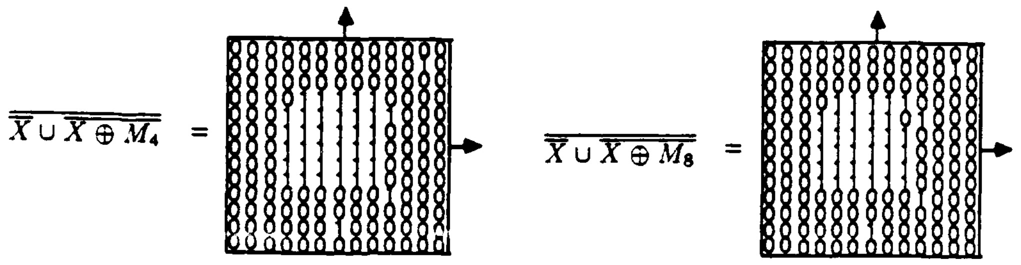

$\overline{\overline{X} \cup \overline{X \oplus M_4}}$ =
$\overline{\overline{X} \cup \overline{X \oplus M_8}}$ =

Figure 11(b): "Pepper" noise removal.

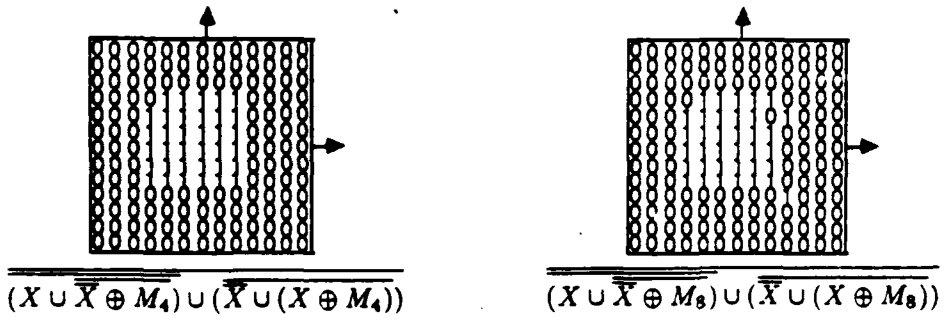
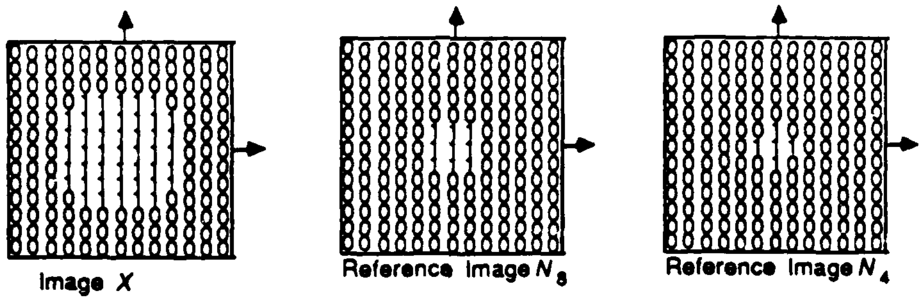
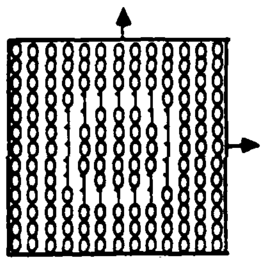

$\overline{(X \cup \overline{\overline{X} \oplus M_4}) \cup (\overline{\overline{X} \cup (X \oplus M_4)})}$
$\overline{(X \cup \overline{\overline{X} \oplus M_8}) \cup (\overline{\overline{X} \cup (X \oplus M_8)})}$

Figure 11(c): "Salt" and "pepper" noise removal.



$X/(X \ominus N_4) =$



$X/(X \ominus N_8) =$

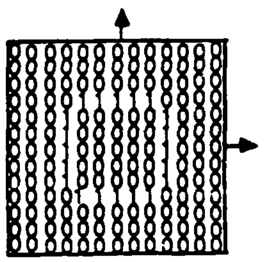


Figure 12: Edge Detection.

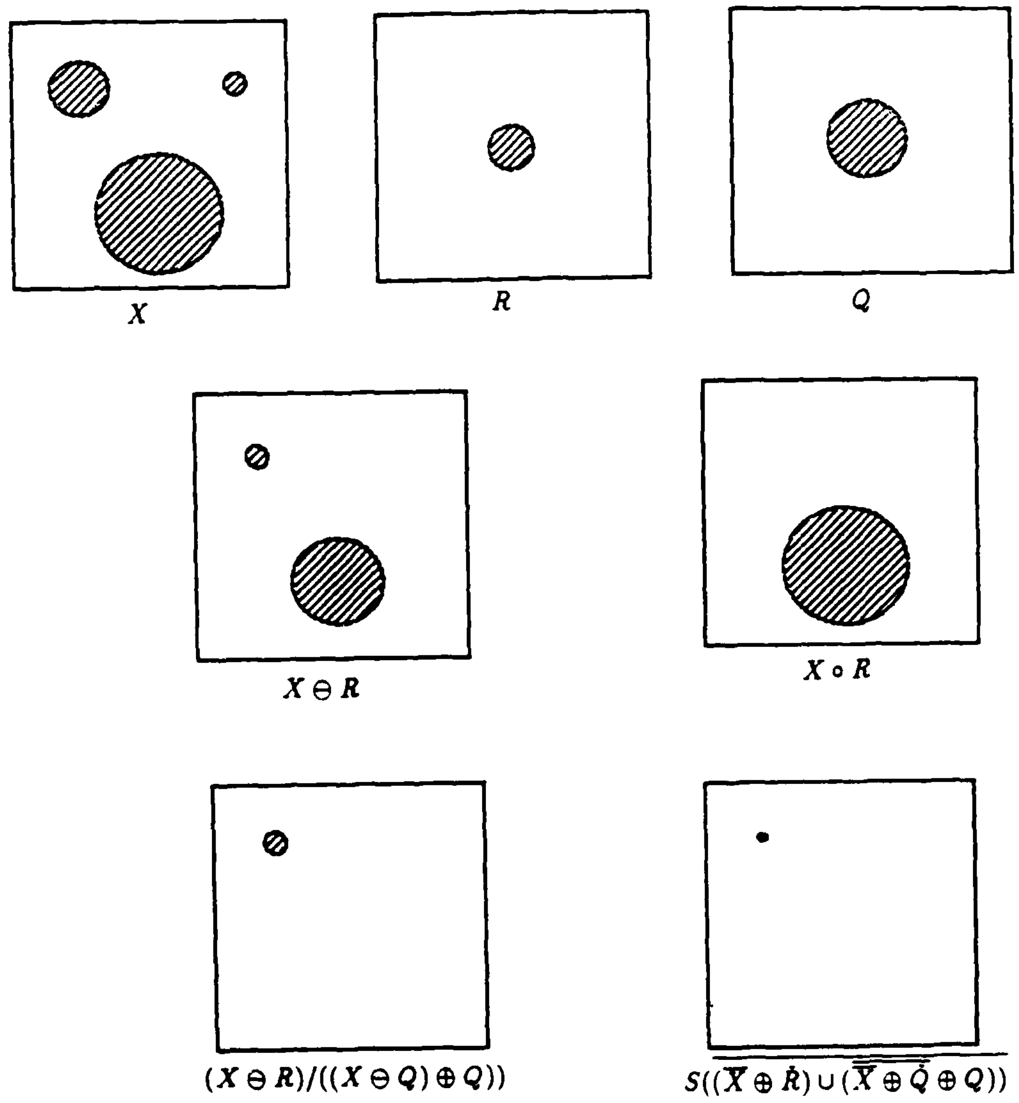

X
R
Q
$X \ominus R$
$X \circ R$
$(X \ominus R)/((X \ominus Q) \oplus Q))$
$S(\overline{(\overline{X} \oplus \dot{R}) \cup (\overline{\overline{X} \oplus \dot{Q}} \oplus Q)})$

Figure 13: A size verification (for holes).

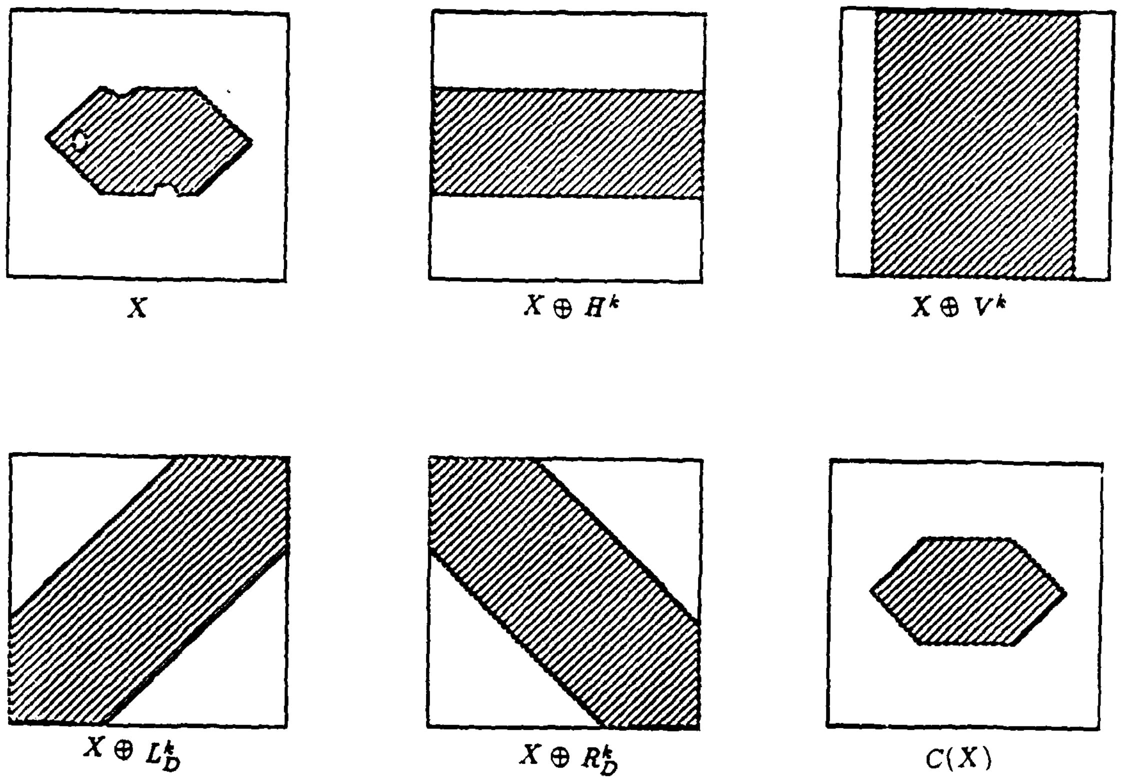


Figure 14(a): An example of the convex hull of an image X (implemented by the intersection of projections).

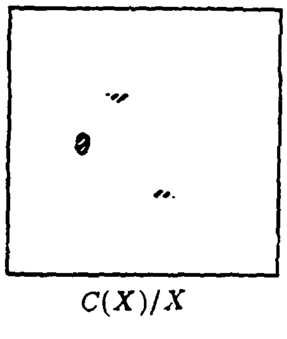


Figure 14(b): The difference of $C(X)$ by X.

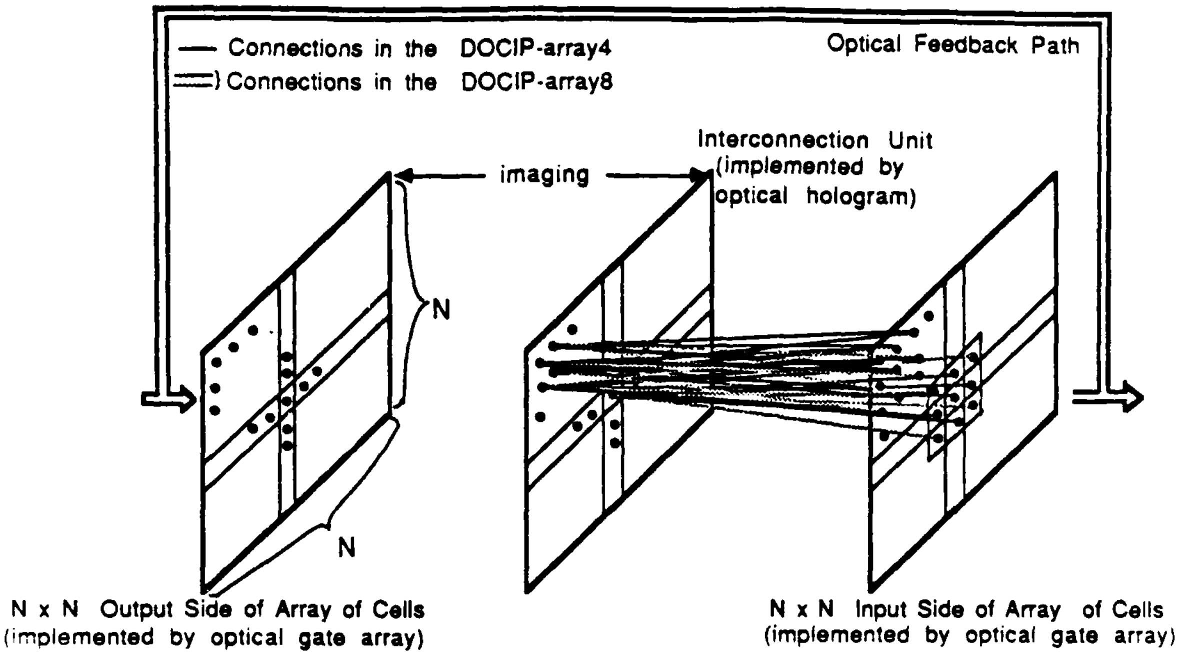

— Connections in the DOCIP-array4
═) Connections in the DOCIP-array8
Optical Feedback Path
Interconnection Unit (implemented by optical hologram)
imaging
N
N
N x N Output Side of Array of Cells (implemented by optical gate array)
N x N Input Side of Array of Cells (implemented by optical gate array)

Figure 15: An optical 4-connected or 8-connected cellular array (DOCIP-array4 or DOCIP-array8). Imaging optics are omitted for clarity. Each cell connects with its four nearest cells and itself by optical 3-D free interconnection. The input and output sides of the optical gate array are interconnected by an optical feedback path and are shown separately for clarity.

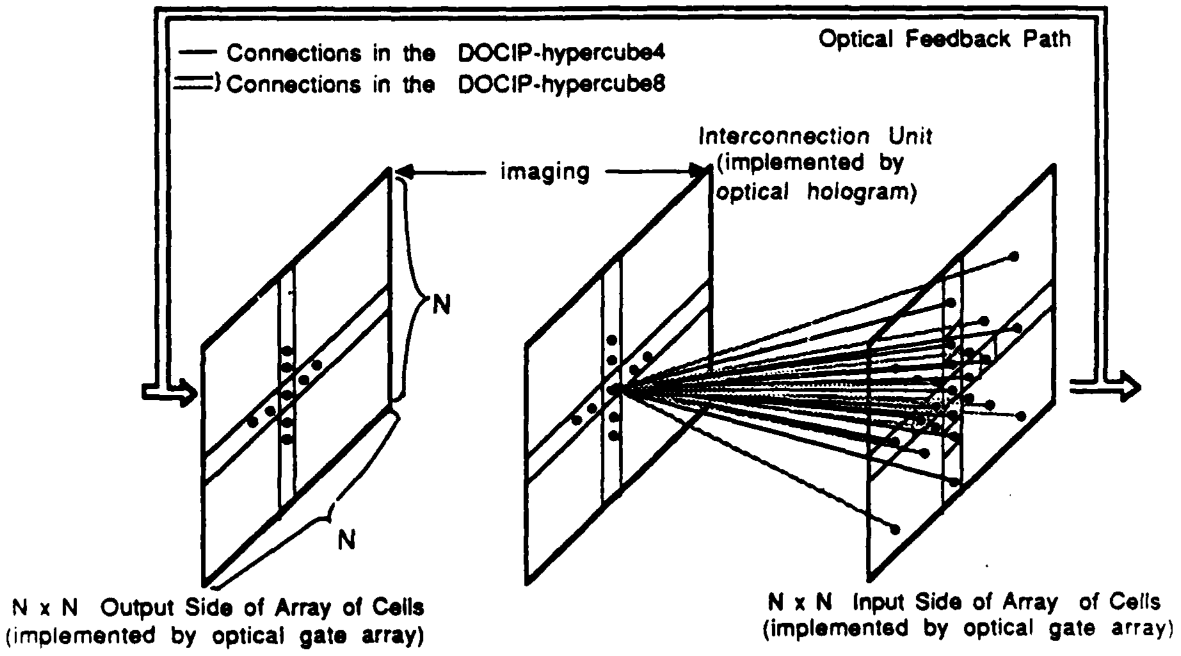

— Connections in the DOCIP-hypercube4
═) Connections in the DOCIP-hypercube8
Optical Feedback Path
Interconnection Unit (implemented by optical hologram)
imaging
N
N
N x N Output Side of Array of Cells (implemented by optical gate array)
N x N Input Side of Array of Cells (implemented by optical gate array)

Figure 16: An optical 4-directed or 8-directed cellular hypercube (DOCIP-hypercube4 or DOCIP-hypercube8). Each cell connects with cells in the 4 directions or 8 directions at distances $1, 2, 4, 8, ..., 2^k$ from it by optical 3-D free interconnection.

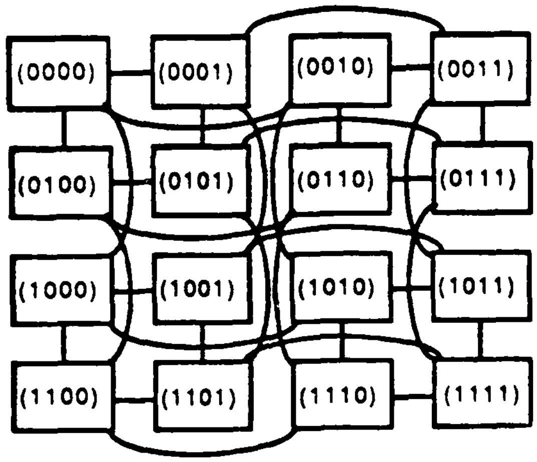


Figure 17: A conventional hypercube (4-cube) laid out in two dimensional space. Its interconnections have no spatial invariance.

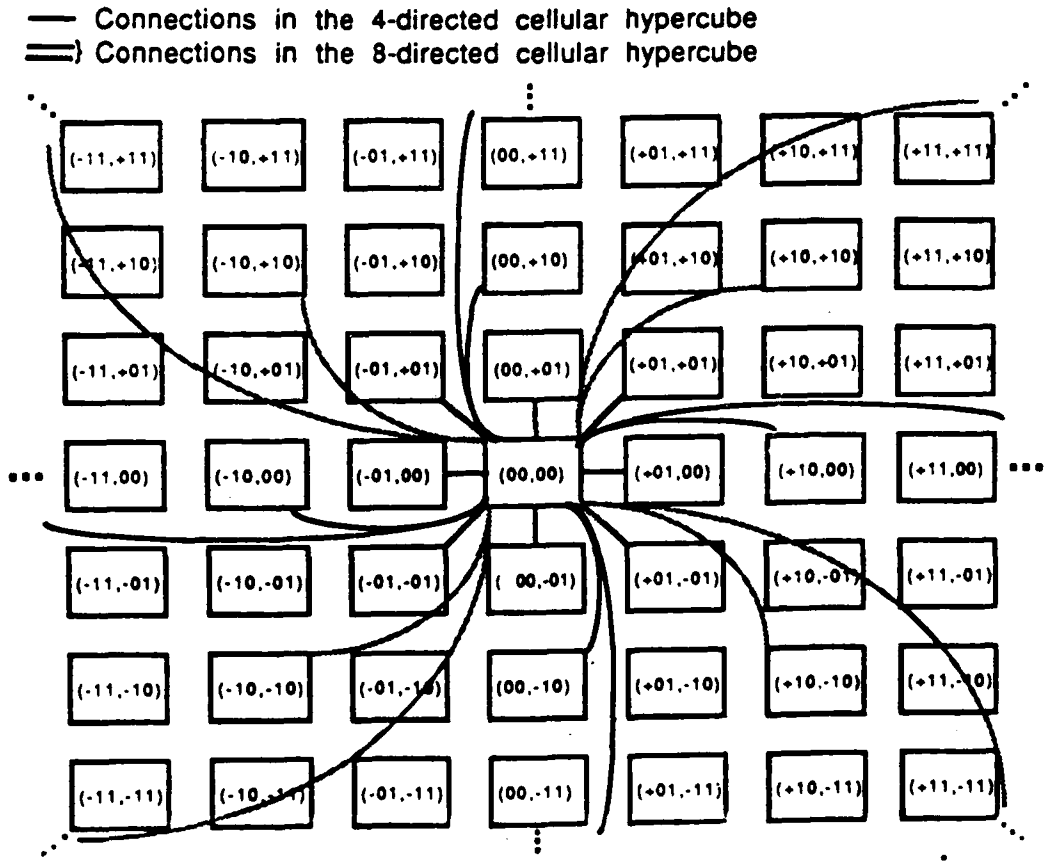


Figure 18: A two-dimensional cellular hypercube — DOCIP-hypercube. Each cell is interconnected with other cells having a *relative* one bit difference in coordinate label in positive or negative x and y directions to achieve a spatially symmetric and invariant interconnection pattern. Only connections from the central cell are shown; all cells are connected identically so the resulting interconnections are space invariant.

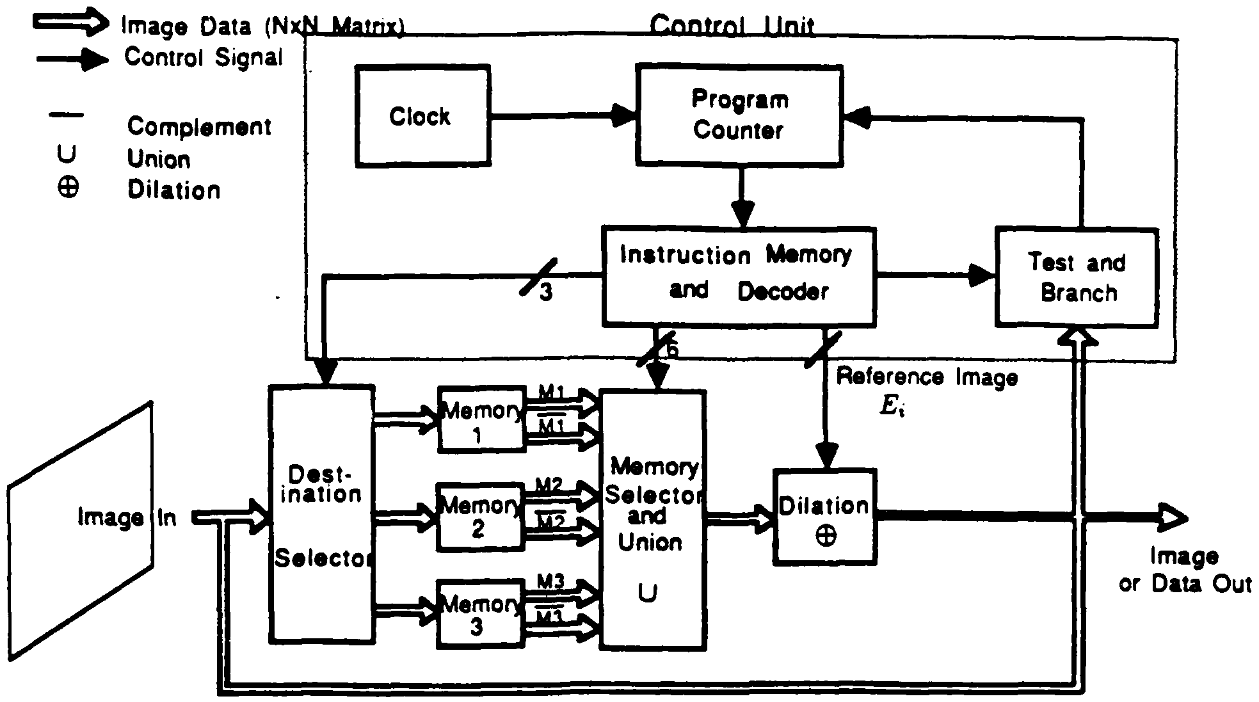


Figure 19: A digital optical cellular image processor (DOCIP) architecture — one implementation of binary image algebra (BIA). The DOCIP-array requires 9 (or 5) control bits for reference image E_i. The DOCIP-hypercube requires $O(logN)$ control bits for reference image E_i.

TECHNOLOGY / OPERATION	Conventional Array (Electronics)	DOCIP-array (Optics)	DOCIP-hypercube (Optics)
Local Operations	O(1)	O(1)	O(1)
Global Operations	O(N) or O(N^2)	O(N)	O(logN)
Communication PE↔Main Memory	O(N) or O(N^2)	O(1)	O(1)
Input/Output	O(N) or O(N^2)	O(1)	O(1)

Table 2: Cellular image processor execution times for $N \times N$ image data. It roughly compares the execution time for the conventional electronic array processor, the DOCIP-array and the DOCIP-hypercube.

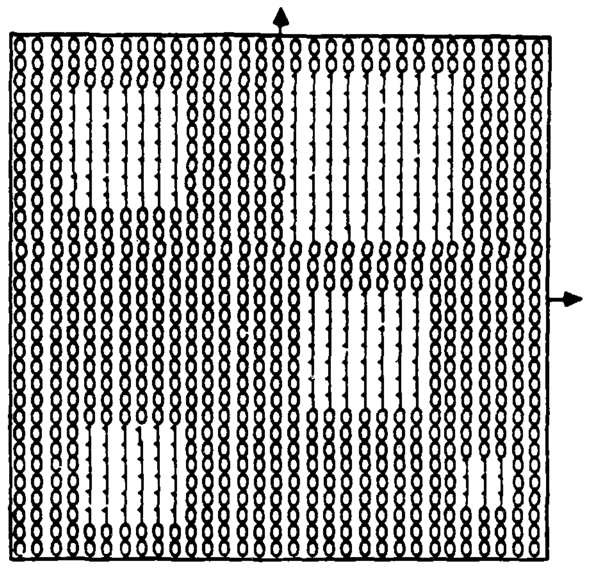
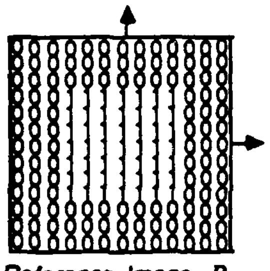


Figure 20: The input image X.



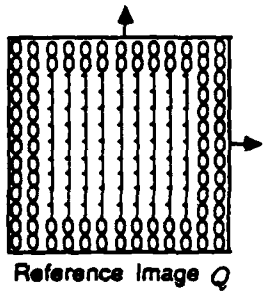


Figure 21: The reference images R and Q.

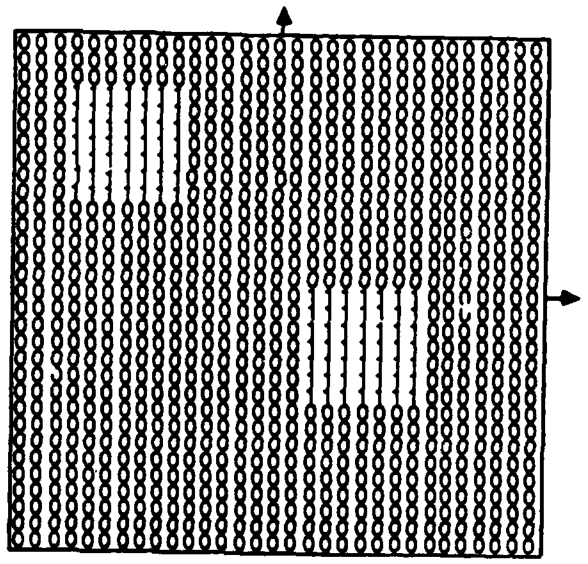


Figure 22: The expected output image Y.

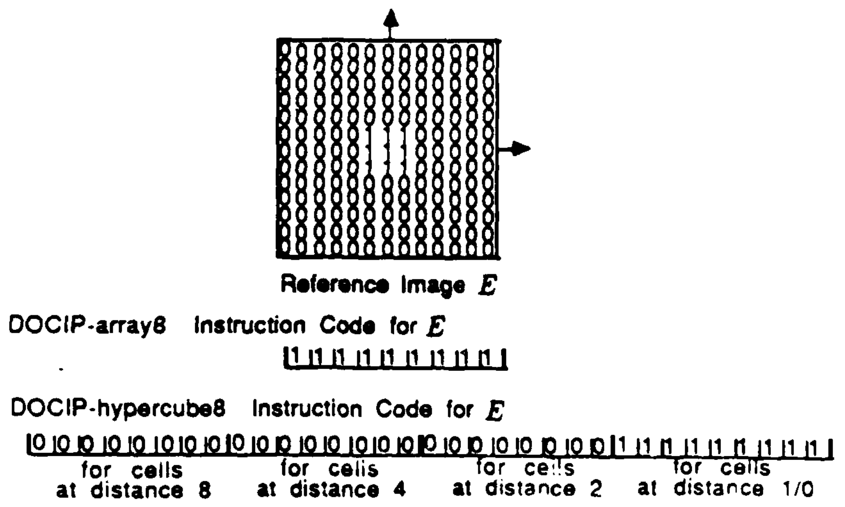

Reference Image E
DOCIP-array8 Instruction Code for E
DOCIP-hypercube8 Instruction Code for E
for cells at distance 8
for cells at distance 4
for cells at distance 2
for cells at distance 1/0

Figure 23: An allowed reference image E at a clock cycle in the DOCIP-array8 (also allowed in DOCIP-hypercube8) and its corresponding 9 (or 33) bits in instruction $(n_1 n_2 ... n_9)$ for controlling the neighborhood mask (i.e. the reference image for the dilation).

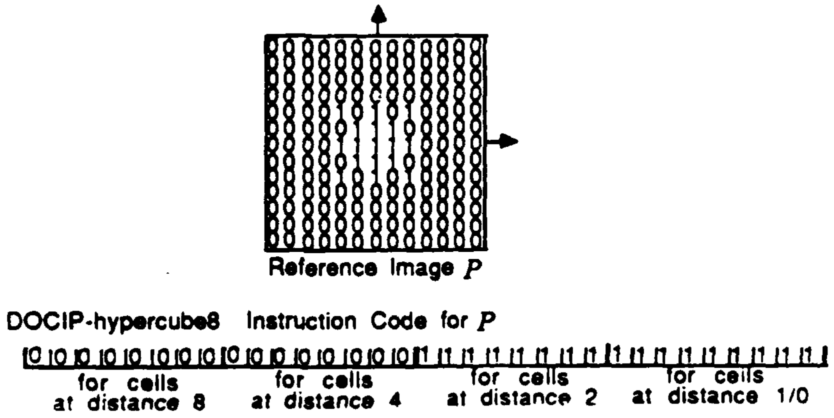


Figure 24: An allowed reference image P at a clock cycle in the DOCIP-hypercube8 (not allowed in the DOCIP-array8) and its corresponding 33 bits (assume 31×31 cells) in instruction $(n_1 n_2 ... n_{33})$ for controlling the neighborhood mask (i.e. the reference image for the dilation).

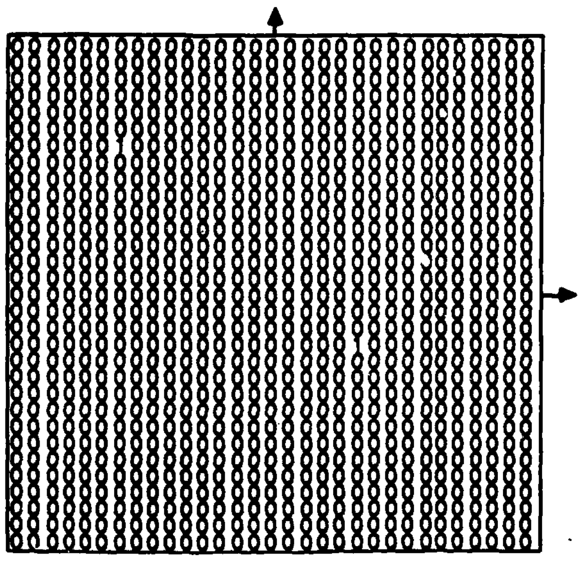


Figure 25: The locations of the desired objects in the output image Y.

Optical Symbolic Substitution and Pattern Recognition Algorithms Based on Binary Image Algebra

K. S. Huang, B. K. Jenkins, A. A. Sawchuk

Signal and Image Processing Institute
Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089-0272, USA

February 27, 1988

Abstract

Pattern recognition algorithms and algebraic properties of binary image algebra (BIA) are used to improve the speed, flexibility and complexity of symbolic substitution.

Submitted to *ICO Topical Meeting On Optical Computing*, Toulon, France, August 29 - September 2, 1988.

Optical Symbolic Substitution and Pattern Recognition Algorithms Based on Binary Image Algebra

K. S. Huang, B. K. Jenkins, A. A. Sawchuk
Signal and Image Processing Institute, Department of Electrical Engineering,
University of Southern California, Los Angeles, CA 90089-0272, USA

Summary

Binary image algebra (BIA), a unified systematic complete theory of parallel binary image processing [1], also provides a unified spatial logic of digital optical computing for describing symbolic substitution, cellular logic and Boolean logic in parallel [2]. Symbolic substitution has been used to implement logic, arithmetic, communication and simulating a Turing machine [3]; but its implementation of some operations (e.g. parallel binary arithmetic) is relatively complicated to other BIA implementations [2]. In this paper we further suggest some BIA algebraic techniques and pattern recognition algorithms, including a shift, scale and rotation invariant algorithm, to improve the speed, flexibility and complexity of symbolic substitution.

A symbolic substitution rule involves two steps: 1) recognizing the locations of a certain spatial search-pattern within the 2-D input data, and 2) substituting a new replacement-pattern wherever the search-pattern is recognized. As illustrated in Fig. 1, BIA can be used to realize a symbolic substitution rule defined by:

$$(X \circledast R) \oplus Q = ((X \ominus R_1) \cap (\overline{X} \ominus R_2)) \oplus Q = \overline{(\overline{X} \oplus \check{R}_1) \cup (X \oplus \check{R}_2)} \oplus Q \qquad (1)$$

where X is the 2-D input data, $R = (R_1, R_2)$ is the reference image pair corresponding to the search-pattern (R_1 and R_2 define the foreground and the background of the search-pattern respectively), $\check{R}$ defines a reflected reference image given by $\check{R} = \{(-x, -y) \mid (x, y) \in R\}$, Q is the reference image corresponding to the replacement-pattern, "$\circledast$" denotes the hit or miss transform which is the pattern recognizer, "$\ominus$" denotes the erosion operation, and "$\oplus$" denotes the dilation operation which is the pattern replacement operator. To work with more than one rule (say p substitution rules) for practical applications, a symbolic substitution system (Fig. 2) produces several copies of the input X, provides p different recognizer-substituter units, and then combines the outputs of various units to form a new output. Thus, a symbolic substitution system is implemented by

$$\bigcup_{i=1}^{p} (X \circledast R^{(i)}) \oplus Q^{(i)} \qquad (2)$$

where $R^{(i)}$ and $Q^{(i)}$, $i = 1, 2, ..., p$, are the reference image pairs and replacement patterns in the i^{th} symbolic substitution rule. This, then, is the BIA formula for general symbolic substitution.

However, in many cases the above form is inefficient and can be reduced to a relatively simpler form or implemented in a more efficient way by using some BIA algebraic techniques. Here are some examples: 1) the full recognition can be implemented by only the background or foreground recognition under certain conditions; 2) if $Q^{(i)} = \phi$, the i^{th} symbolic substitution rule in Eq. (2) is not needed (e.g. the four rules of binary subtraction in simple intensity coding of arithmetic data can be reduced to only two rules [2]); and 3) if $Q^{(i)} = Q$ for all $1 \le i \le p$ (this happens in those cases that a class of search-patterns is defined by a set of reference image pairs $R^{(i)}$, $i = 1, 2, ..., p$), we should combine the results of the hit or miss transforms first and then replace them by the same replacement-pattern Q instead of implementing p substitution units for realizing the same substitution step, i.e.

$$(\bigcup_{i=1}^{p} X \circledast R^{(i)}) \oplus Q. \qquad (3)$$

The practical difficulty with the implementation in Eqs. (2) and (3) is that the hit or miss transform is only efficient for the shift invariant recognition and would require a large number of intricate reference image pairs to perform the recognition step in the presence of changes in scale, rotation or both. Thus, it might be too costly to implement scale and rotation invariant recognition of intricate patterns for symbolic substitution based on the above formula. For example, if we want to substitute all "square patterns" in an input image by the same character "S", it would be very inefficient to use the above symbolic substitution implementation techniques.

To solve this kind of scale and rotation invariant problem, here we recognize all the desired patterns by reversing the growing procedure of a family of patterns. This family defines all patterns in the presence of changes in scale, rotation or both, and transforms all the desired patterns into their original seeds, which are isolated single image points. We have developed a description of this procedure in terms of BIA. For brevity, here we describe only the case of shift and scale invariant recognition. Suppose we want to recognize all square patterns with different scales and locations in the input image X (e.g. Fig. 3(a)) and to produce the output image Y (e.g. Fig. 3(b)). The procedure is: 1) determine a growing sequence of the desired patterns T_i (e.g. Fig. 3(c)), where $0 \le i \le m$ and the largest size of the desired patterns is $m \times m$; 2) find a small set of *good* reference image pairs $\{R(\theta)\}$ (e.g. Fig. 3(d) has only 5 small reference image pairs for recognizing all square objects with different scales) satisfying some criteria, where each reference image pair in $\{R(\theta)\}$ corresponds to a possible neighborhood of a given *foreground* image point in a pattern T_i, $1 \le i \le m$, whose previous state in the pattern T_{i-1} is a *background* point; 3) transform the desired patterns T_i, $i = 1, 2, ..., m$, in the 2-D input image $X = X(t_0)$ into their original seeds (i.e. T_0 which contains one and only one foreground image point) by the recursive relation $X(t_{k+1}) = X(t_k) / \bigcup_{\theta \in \Theta} X(t_k) \circledast R(\theta)$, where $0 \le k \le m$; and 4) pick up the original seeds by $Y = X(t_m) \circledast Q$, where Q (Fig. 3(e)) is a *reference image pair with one and only one foreground image point at the center and* Y is the final recognition output. By selecting *good* reference image pairs associated the growing sequences of rotation patterns, we can extend shift and scale invariance to include rotation invariance in a similar way. This algorithm can efficiently reduce the computation complexity for a certain class of pattern recognition and symbolic substitution problems; their computation times depend only on the diameter of the largest desired pattern, but not on the number of patterns nor the size of the whole image.

A digital optical cellular image processor (DOCIP) [1] [2] implements all the above algorithms of symbolic substitution and pattern recognition in a flexible and efficient way compared to a symbolic substitution processor (Fig. 2) with p fixed recognizer-substituter units. The DOCIP programming for these algorithms will be illustrated.

References

[1] K. S. Huang, B. K. Jenkins, A. A. Sawchuk, "Binary Image Algebra and Optical Cellular Logic Processor Design," submitted to *Computer Vision, Graphics, and Image Processing*.

[2] K. S. Huang, B. K. Jenkins, A. A. Sawchuk, "An Image Algebra Representation of Parallel Optical Binary Arithmetic", submitted to *Applied Optics*.

[3] K.-H. Brenner, A. Huang, and N. Streibl, "Digital Optical Computing with Symbolic Substitution," *Applied Optics*, Vol. 25, pp. 3054-3060, 1986.

Acknowledgements

This research was supported by the Air Force Office of Scientific Research under grant AFOSR-84-0181, by the Office of Naval Research under contract N00014-86-K-0602, and by an IBM graduate fellowship.

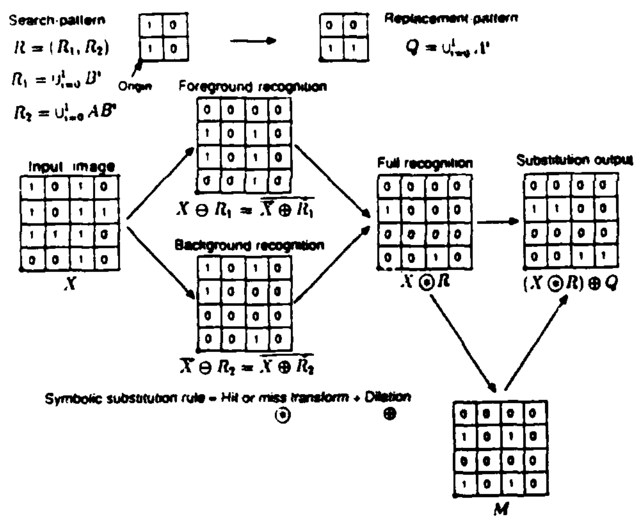
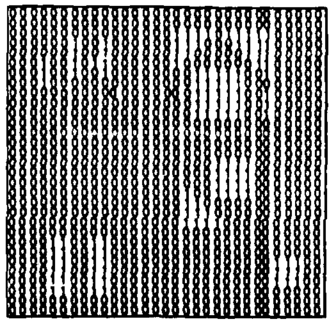
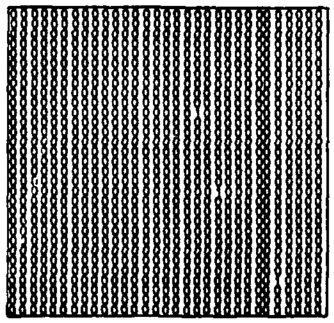
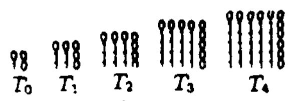
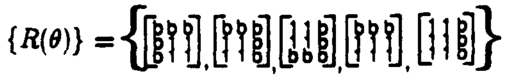

Search pattern
$R = (R_1, R_2)$
$R_1 = \cup_{i=0}^{1} B^i$
$R_2 = \cup_{i=0}^{1} AB^i$
Origin
Replacement pattern
$Q = \cup_{i=0}^{1} A^i$
Foreground recognition
Input image
X
$X \ominus R_1 = \overline{\overline{X} \oplus \check{R}_1}$
Background recognition
$\overline{X} \ominus R_2 = \overline{X \oplus \check{R}_2}$
Full recognition
$X \circledcirc R$
Substitution output
$(X \circledcirc R) \oplus Q$
Symbolic substitution rule = Hit or miss transform + Dilation
M

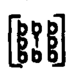
Figure 1. BIA representation of symbolic substitution. The optional mask M is for controlling the block search region.


(a) An input image X.


(b) The output image Y.


(c) The growing sequence of square patterns T_i, $o \le i \le 4$.


(d) A set of *good* reference image pairs $\{R(\theta)\}$ for square patterns with different scales.


1: foreground points with value 1
b: background points with value 0

(e) The reference image pair Q.

Figure 3. A shift and scale invariant pattern recognition of square patterns.

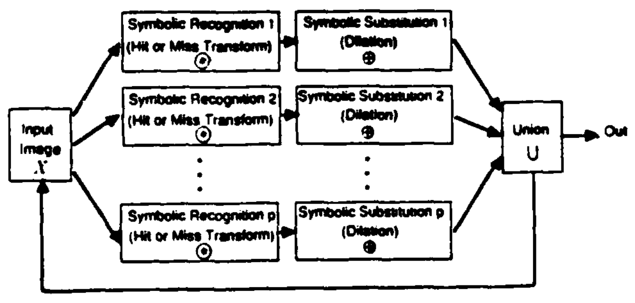

Input Image X
Symbolic Recognition 1 (Hit or Miss Transform)
Symbolic Substitution 1 (Dilation)
Symbolic Recognition 2 (Hit or Miss Transform)
Symbolic Substitution 2 (Dilation)
Symbolic Recognition p (Hit or Miss Transform)
Symbolic Substitution p (Dilation)
Union
Out

Figure 2. A symbolic substitution system with p symbolic substitution rules.

2.2 Digital Optical Cellular Architectures

The papers reprinted in this section discuss details of optical cellular architectures and their instruction set.

The DOCIP is a 2-D, page oriented array of individual processors located at every pixel of a large image. The attached paper by K.S. Huang, B.K. Jenkins and A.A. Sawchuk, "Binary Image Algebra and Optical Cellular Logic Processor Design", submitted to *Computer Vision, Graphics and Image Processing*, summarizes some of these concepts and their algebraic background. Following this paper is "Optical Symbolic Substitution and Pattern Recognition Algorithms Based on Binary Image Algebra", by K.S. Huang, B.K. Jenkins and A.A. Sawchuk, from the *ICO Topical Meeting on Optical Computing,* Toulon, France, 1988, which contains additional information.

This paper is concerned with the hardware implementation of one cell of a prototype digital optical cellular image processor (DOCIP).

Implementation of
A Prototype Digital Optical Cellular Image Processor (DOCIP)

K. S. Huang, A. A. Sawchuk, B. K. Jenkins, P. Chavel*, J. M. Wang*, A. G. Weber, C. H. Wang, I. Glaser

Signal and Image Processing Institute
Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089-0272, USA

*Institut d'Optique
Laboratoire Associé au CNRS
Université de Paris Sud
BP43, 91406 Orsay cedex, FRANCE

February 27, 1988

Abstract

A processing element of a prototype digital optical cellular image processor (DOCIP) is implemented to demonstrate a particular parallel computing and interconnection architecture.

Submitted to *ICO Topical Meeting On Optical Computing*, Toulon, France, August 29 - September 2, 1988.

Implementation of A Prototype Digital Optical Cellular Image Processor (DOCIP)

K. S. Huang, A. A. Sawchuk, B. K. Jenkins, P. Chavel*, J. M. Wang*, A. G. Weber, C. H. Wang, I. Glaser

Signal and Image Processing Institute, Department of Electrical Engineering,
University of Southern California, Los Angeles, CA 90089-0272, USA

*Institut d'Optique, Laboratoire Associé au CNRS,
Université de Paris Sud, BP43, 91406 Orsay cedex, FRANCE

Summary

Digital optical cellular image processor (DOCIP) architectures, DOCIP-array and DOCIP-hypercube, can perform the tasks of parallel binary image processing and parallel binary arithmetic [1]. The use of optical interconnections permits a cellular hypercube topology to be implemented without paying a large penalty in chip area (the cellular hypercube interconnections are space-invariant which implies relatively low hologram complexity); it also enables images to be input to and output from the machine in parallel. Table 1 gives a comparison of three different interconnection networks: cellular array (DOCIP-array interconnection network), conventional hypercube, and cellular hypercube (DOCIP-hypercube interconnection network). In this paper we experimentally demonstrate the concept of the DOCIP architecture by implementing one processing element of a prototype optical computer including a 49-gate processor, an instruction decoder, and electronic input/output interfaces.

A multiple-exposure multi-facet interconnection hologram provides the fixed interconnections between the outputs and the inputs of an array of 7 × 7 optical gates. The input data and the instructions are supplied from an LED array. The outputs of optical gates are detected by a video camera and compared with the results of a software simulation. A diagram of the main components of this experimental system is shown in Fig. 1.

A space-variant interconnection system [2] for within-processor interconnection is used in this experimental demonstration. A computer controlled system is used to make an array of 49 interconnection subholograms. An optical point source S, whose position is controlled by the mirror M2 with two rotational stages (Fig. 1), is used to provide an object beam for determining an interconnection of a subhologram in the multi-facet hologram. A mask with a circular aperture, controlled by two translational stages, is used to determine the sizes and positions of subholograms in a holographic plate. The interconnection hologram for this 49-gate optical processing element comprises 49 subholograms, which are laid out in a 7 × 7 array. Each subhologram covers a circular area with a diameter of 1.5 mm. The spacing between the centers of two subholograms is 3.0 mm. Note that the path of the object beam and the mask for subholograms are only used for making the interconnection hologram; they are blocked or moved when we reconstruct the hologram to implement the interconnections of the optical gates. We use a volume phase hologram with a dichromated gelatin medium for obtaining high diffraction efficiencies.

The array of 7 × 7 optical gates is implemented by a Hughes liquid-crystal light valve (LCLV) with liquid-crystal molecules in a 45° twisted nematic configuration [2]. The LCLV is read out between crossed polarizers and is biased to implement a NOR operation. The gate size in this experiment has a diameter of 0.3 mm and the spacing between the centers of two gates is 0.6 mm.

The circuit diagram of the processing element, as shown in Fig. 2, consists of 49 NOR gates with maximum fan-in of 3 and fan-out of 4. The processing element includes a 3-bit destination selector, a 3-bit master-slave flip-flop memory, a 6-bit memory selector with a union module, and a 5-bit neighborhood selector (for DOCIP-array4 [1]) with a dilation module. This experimental DOCIP system has one instruction, supplied from an LED array and decoded by the optical hardware. This instruction has the format: $(c, d_1, d_2, d_3, s_1, s_2, ..., s_6, n_1, n_2, ..., n_5)$ where c selects the image from the input or from the feedback; d_1, d_2, and d_3 select the destination memory for storing the image; $s_1, s_2, ..., s_6$ select the output from the memory elements; and $n_1, n_2, ..., n_5$ control the neighborhood mask, i.e. supply the reference image. We will experimentally demonstrate the DOCIP architecture concept with this system.

References

[1] K. S. Huang, B. K. Jenkins, A. A. Sawchuk, "Binary Image Algebra and Optical Cellular Logic Processor Design," submitted to *Computer Vision, Graphics, and Image Processing*; K. S. Huang, B. K. Jenkins, A. A. Sawchuk, "An Image Algebra Representation of Parallel Optical Binary Arithmetic", submitted to *Applied Optics*.

[2] A. A. Sawchuk and T. C. Strand, "Digital Optical Computing," *Proc. IEEE*, Vol.72, pp. 758-779,1984; B. K. Jenkins, et al, "Sequential Optical Logic Implementation," *Applied Optics*, Vol. 23, No. 19, pp. 3455-3464, 1984; B. K. Jenkins, et al, "Architectural Implications of A Digital Optical Processor," *Applied Optics*, Vol. 23, No. 19, pp. 3465-3474, 1984.

Acknowledgements

The authors thank W. Kressen, K. Prager and T. Mayeda for their contributions to this work. This research was supported by the Air Force Office of Scientific Research under grant AFOSR-84-0181 and by an IBM graduate fellowship.

Interconnections	Cellular Array	Conventional Hypercube	Cellular Hypercube
Connectivity (Interconnections per PE)	O(1)	O(logN)	O(logN)
2-D Spatial Invariance	Yes	No	Yes
Inter-PE Communication Time Complexity	O(N)	O(logN)	O(logN)
VLSI Chip Area	$O(N^2)$	$O(N^4)$	$O(N^4)$
Hologram Space-bandwidth Product	$O(N^2)$	$O(N^2 \log N)$	$O(N^2)$

Table 1. A comparison between three different interconnection networks of NxN processeng elements (PEs): cellular array, conventional hypercube and cellular hypercube. When laid out on a VLSI chip, both the conventional hypercube and cellular hypercube pay a large penalty in chip area while the cellular hypercube has a relatively low hologram complexity.

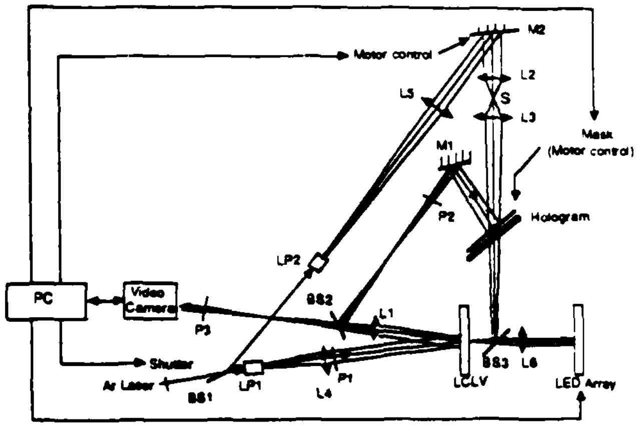


Figure 1. Experimental DOCIP system. Lens L1 images from the LCLV gate output plane to the hologram plane. Beam Splitter BS3 combines the external input signals from *LED* array and the *feedback signals* from *interconnection hologram*. LP1 and LP2 are lens-pinhole assemblies. P1 and P2 are crossed polarizers. The hologram comprises an array of subholograms. Mirror M2 controls the position of point source S during hologram exposure. After the hologram is made, the mask and all components in the path from BS1 to the hologram are not needed.

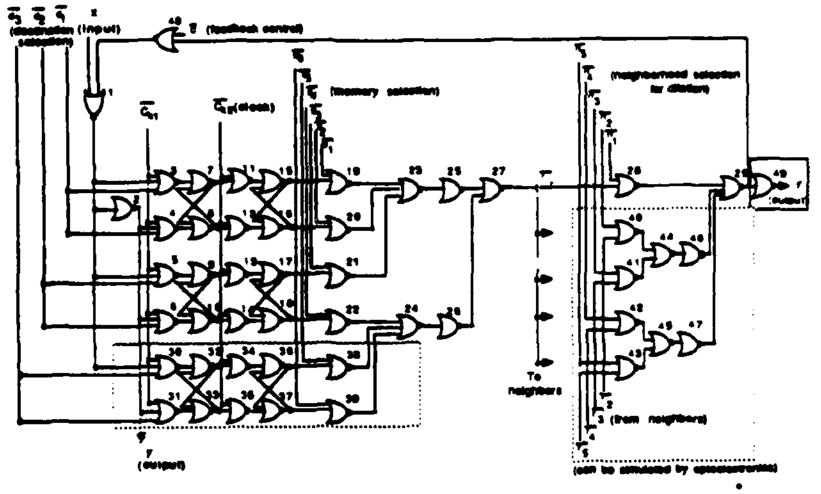


Figure 2. The circuit diagram of a 49-gate processing element of the DOCIP-array4.

3 Written Publications

During the period 1 July 1987 through 30 June 1988, a number of papers based on this research have been published or submitted. A list of these follows:

1. Huang, K.S., Jenkins, B.K. and Sawchuk, A.A., "Binary Image Algebra Representations of Optical Cellular Logic and Symbolic Substitution," *OSA Annual Meeting,* Rochester, NY, October 18-23, 1987.

2. Huang, K.S., Jenkins, B.K. and Sawchuk, A.A., "Programming A Digital Optical Cellular Image Processor," *OSA Annual Meeting,* Rochester, NY, October 18-23, 1987.

3. Huang, K.S., Jenkins, B.K. and Sawchuk, A.A., "Optical Cellular Logic Architectures Based on Binary Image Algebra," *Proc. IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Machine Intelligence,* Seattle, WA, October 1987.

4. Huang, K.S., Jenkins, B.K. and Sawchuk, A.A., "A Cellular Hypercube Architecture for Image Processing," *Applications of Digital Image Processing X,* Proc. of SPIE-The International Society for Optical Engineering, Vol 829, San Diego, CA, August 1987.

5. Huang, K.S., Jenkins, B.K., Sawchuk, A.A., "An Image Algebra Representation of Parallel Optical Binary Arithmetic," submitted to, *Applied Optics.*

6. Huang, K.S., Jenkins, B.K. and Sawchuk, A.A., "Binary Image Algebra and Optical Cellular Logic Processor Design," submitted to, *Computer Vision, Graphics, and Image Processing.*

7. Huang, K.S., Jenkins, B.K. and Sawchuk, A.A., "Optical Symbolic Substitution and Pattern Recognition Algorithms Based on Binary Image Algebra," submitted to, *ICO Topical Meeting on Optical Computing,* Toulon, France, August 29 - September 2, 1988.

8. Huang, K.S., Sawchuk, A.A., Jenkins, B.K., Chavel, P., Wang, J.M., Weber, A.G., Wang, C.H., Glaser, I., "Implementation of A Prototype Digital Optical Cellular Image Processor (DOCIP)," submitted to, *ICO Topical Meeting On Optical Computing,* Toulon, France, August 29 - September 2, 1988.

4 Professional Personnel and Advanced Degrees

The following individuals contributed to the research effort supported by this grant:

Dr. Alexander A. Sawchuk, Professor of Electrical Engineering, Director, Signal and Image Processing Institute; Principal Investigator.

Dr. B.K. Jenkins, Research Assistant Professor of Electrical Engineering; Senior Investigator.

Kung-Shiuh Huang, Research Assistant, Ph.D. Candidate, Department of Electrical Engineering.

Dr. Isaia Glaser, Visiting Associate Professor of Electrical Engineering.

Dr. Pierre Chavel, Visiting Scientist, Signal and Image Processing Institute.

5 Interactions (Coupling Activities)

During the period 1 July 1987 through 30 June 1988, oral presentations have been made at meetings and conferences based on this work. A list of these follows:

1. Huang, K.S., Jenkins, B.K. and Sawchuk, A.A., "Binary Image Algebra Representations of Optical Cellular Logic and Symbolic Substitution," *OSA Annual Meeting,* Rochester, NY, October 18-23, 1987.

2. Huang, K.S., Jenkins, B.K. and Sawchuk, A.A., "Programming A Digital Optical Cellular Image Processor," *OSA Annual Meeting,* Rochester, NY, October 18-23, 1987.

3. Huang, K.S., Jenkins, B.K. and Sawchuk, A.A., "Optical Cellular Logic Architectures Based on Binary Image Algebra," *Proc. IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Machine Intelligence,* Seattle, WA, October 1987.

4. Huang, K.S., Jenkins, B.K. and Sawchuk, A.A., "A Cellular Hypercube Architecture for Image Processing," *Applications of Digital Image Processing X,* Proc. of SPIE-The International Society for Optical Engineering, Vol 829, San Diego, CA, August 1987.

5. Huang, K.S., Jenkins, B.K. and Sawchuk, A.A., "Optical Symbolic Substitution and Pattern Recognition Algorithms Based on Binary Image Algebra," submitted to, *ICO Topical Meeting on Optical Computing,* Toulon, France, August 29 - September 2, 1988.

6. Huang, K.S., Sawchuk, A.A., Jenkins, B.K., Chavel, P., Wang, J.M., Weber, A.G., Wang, C.H., Glaser, I., "Implementation of A Prototype Digital Optical Cellular Image Processor (DOCIP)," submitted to, *ICO Topical Meeting On Optical Computing,* Toulon, France, August 29 - September 2, 1988.