5-31-1991

Androx implementation of nonlinear filter

Keh-Chao Chu
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

Title of Thesis: Andorx Implementation of Nonlinear Filter

Name: Keh-Chao Chu
Master of Science in Electrical Engineering, 1991
Department of Electrical and Computer Engineering

Thesis directed by: Dr. Nirwan Ansari

Nonlinear filters, such as morphological operations, play an important role in image processing. Several software implementations of nonlinear filters are presented in this thesis. There may exist more than one method to implement a specific operation. The methods used in this thesis are chosen to be most suited for the machine used according to its structure. These most suited methods are also generally best for software implementations on other machines as well. The machine used here is the Androx image computing workstation.

Androx Implementation of Nonlinear Filter

by

Keh-Chao Chu

Thesis submitted to the Faculty of the Graduate School of
the New Jersey Institute of Technology in partial fulfillment of
the requirements for the degree of
Master of Science in Electrical Engineering
1991

APPROVAL SHEET

Title of Thesis: Androx Implementation of
Nonlinear Filter

Candidate: Keh-Chao Chu
Master of Science in Electrical Engineering, 1991

Thesis and Abstract Approved by the Examining Committee:

5/22/91

Dr. Nirwan Ansari, Advisor — Date
Assistant Professor
Department of Electrical and Computer Engineering

5/22/91

Dr. Anthony D. Robbi — Date
Associate Professor
Department of Electrical and Computer Engineering

5-22-91

Dr. Edwin S. H. Hou — Date
Assistant Professor
Department of Electrical and Computer Engineering

New Jersey Institute of Technology, Newark, New Jersey.

VITA

Keh-Chao Chu

Date of Birth:

Place of Birth:

Education:

1989-1991	New Jersey Institute of Technology	MSEE
1981-1985	Chung Yuan Christian University	BS

Position Held:

Winbond Electronic Corp.

Assistant Engineer

1987-1989

ACKNOWLEDGEMENTS

I would like to appreciate Dr. Nirwin Ansari for his advice and support of this thesis. His mastery of the subject matter combined with a great deal of sincerity, enthusiasm and dynamism, have made working on this thesis the most wonderful experience.

Many thanks are due to the members of my committee: Dr. Anthony D. Robbi and Dr. Edwin S.H. Hou.

Also, I would like to thank my wife, Linshiu, and my parents, for their deep loves.

Contents

List of Figures

Chapter 1

Introduction

The Androx is a general purpose and programmable image computer, whose architecture is based on parallel processing. In our system, it is a single-board coprocessor for a Sun Microsystems workstation. With the extensive library of C language callable subroutines for image computations, many of image processing tasks can be implemented in the Androx. Among these tasks, implementing nonlinear filters is the most challenging. Thus, this thesis will mainly describe how to implement nonlinear filters in Androx.

The principal motivation for filtering is the selective removal and attenuation of specific characteristics of signals or images with the intent of making the information within the signal or image more apparent. All of the filters presented herein are based on mathematical morphology. As with any filtering technique, in order to apply a morphological procedure one must know what type of information in the signal is of significance and what characteristics are of no relevance. In morphological filtering, the geometric nature of

the signal or image is of consequence.

The morphological approach is generally based on the analysis of an image in terms of some predetermined geometric shape known as a structuring element. With a judicious choice of a structuring element based on a *priori* information, the output that results when the observed image or signal is acted on by the structuring element will contain more beneficial features and fewer undesirable ones than will the input.

With the availability of parallel processing, some implementations of the basic morphological operations on conventional computers may not be well suited to the machine used here. Because the memory on a general purpose image processing computer is byte addressable, some implementations based on the threshold decomposing bit operation may not be suited to this kind of machine. All of these problems will be addressed in the following chapters.

A brief overview of this thesis is as follows. Chapter 2 introduces the image computing workstation (e.g. the Androx), on which all implementation in this thesis is based. Chapter 3 describes the binary morphological operations. Minkowski operation is first introduced. Binary dilation, erosion, opening, and closing operations are defined in terms of the Minkowski operation. The desired parallel implementation is compared to the implementation on the conventional computer at the end of the chapter. With the definition of medial axis in chapter 4, two thinning algorithms are compared. Chapter 5 deals with gray scale dilation, erosion, opening, and closing op-

erations. The unsuitablity of the threshold decomposing implementation is discussed. Finally, the conclusion is drawn in Chapter 6.

Chapter 2

Androx Image Processing Work Station

The Androx ICS (Image Computing System) consists of four parallel programmable image array processors featuring multiple digital signal processing chips ($DSP's$) with cache memories and a graphics processing chip (GRP) [1]. The programmability of the $DSP's$ provides the flexibility to address the broad spectrum of image processing applications and parallel processing capabilities resulting in system performance. The GRP comes with an extensive library of $C-callable$ graphics processing functions, which allow the user to create attractive image processing displays. In this chapter, we will discuss the system structure, memory configurations, event scheduling, and libraries.

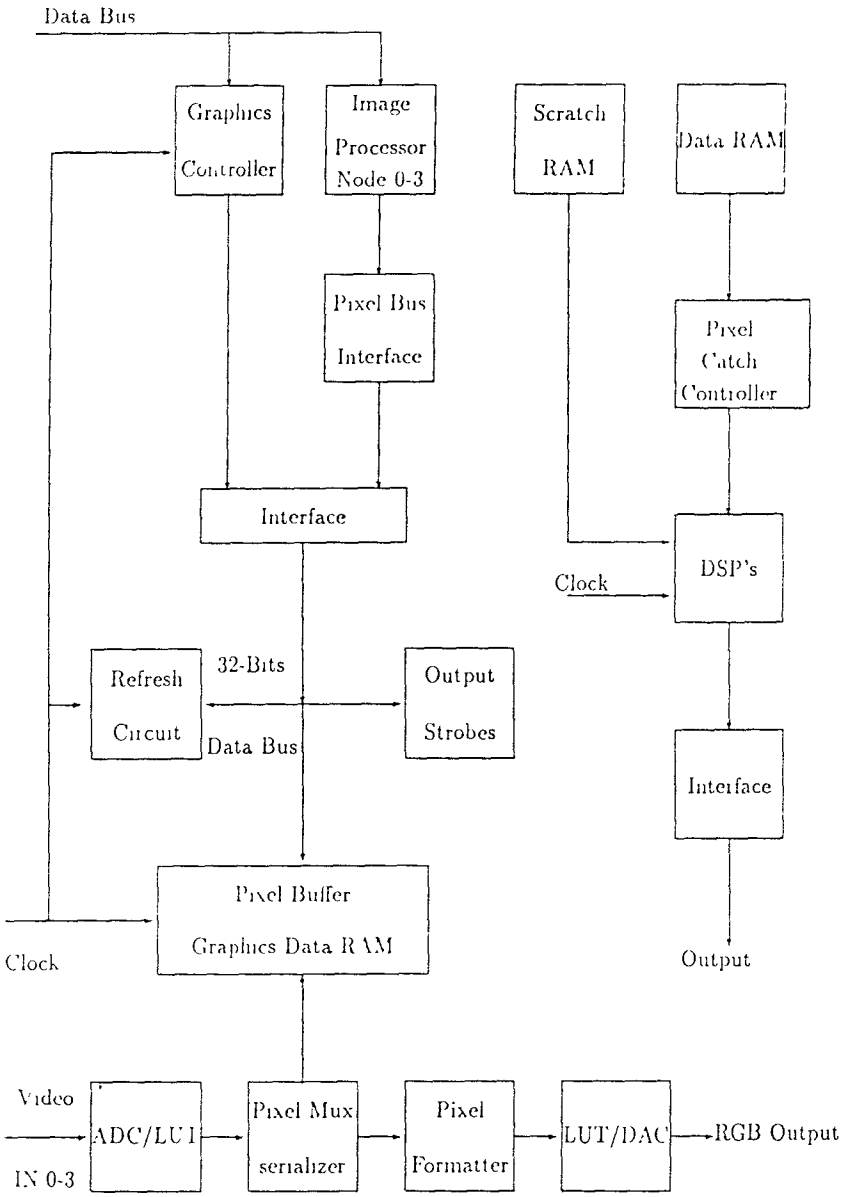


Figure 2.1: The Androx system structure

2.1 System Structure

Figure 2.1 shows the block diagram of the Androx system structure. In this structure, the graphics controller is used to control and transmit graphics data to graphics data RAM. There are four image processing nodes in the system, each node consisting of an analog device ADS2100 and associated cache memory. Image pixels can be transmitted by way of the pixel bus interface to the pixel buffer. The analog-to-digital Converter (ADC) can convert the video signal into pixel data. The pixel data can be processed in look-up tables (LUTs) before being transmitted to pixel serializer and formatter. The formatted pixel data can be transmitted to a digital-to-analog Converter (DAC) for output. The DSP's can read and process the image data from scratch RAM or data RAM, and then output the data through data bus interface.

2.2 Memory Configurations

The Androx $ICS-400XM9$ system has two types of memories: video and scratchpad . The video memory has eight megabytes, which can be addressed as 4096 rows of 2048 bytes. The eight megabytes are equally divided into four banks. When displaying the contents of this video memory in RGB mode, the four banks are assigned specific color planes:

Bank Color

0	Red
1	Green
2	Blue
3	Overlay

The three planes of RGB image must be placed in the same relative location within their respective banks. The scratchpad memory has one megabyte of memory, which can be addressed as 512 rows of 2048 byte.s The lower half-megabyte of scratchpad memory is reserved by the graphics processor. Figure 2.2 illustrates the memory mapping of $ICS - 400XM9$.

2.3 Event Scheduling

If the host processor program invokes the image processor or graphics processor functions, the ICS generally will execute these functions asynchronously. This means that they will continue to execute once they have invoked ICS functions.

To enable the user to synchronize function execution, ICS functions return event numbers, which can be used as input to other functions. The event number can be specified as input to the subsequent function. This event number's flag is cleared only when the function is completed.

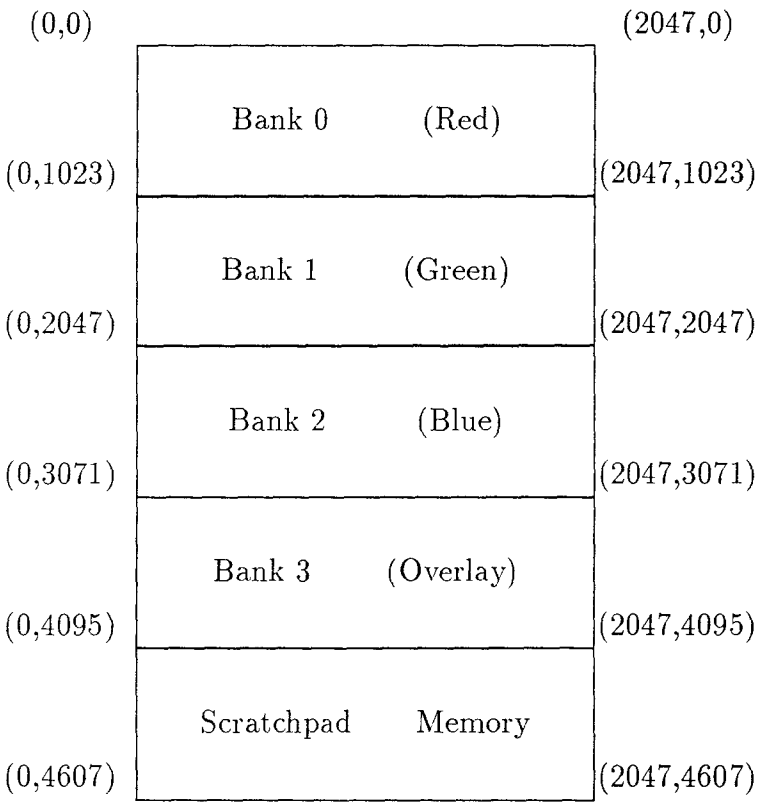


Figure 2.2: The memory mapping of ICS-400XM9

2.4 Androx Libraries

There are four types of libraries from which the user can program his or her applications: applications, graphics, image processing, and video.

The applications library contains some useful functions that simplify certain aspects of writing a program. Some of functions in this library are built out of lower-level library functions, and can be written by the user. The graphics processor can execute only one graphics function at a time. The image processing library routines can return one-dimensional data or two-dimensional data to global memory. The global memory consists of two sections: one used for image acquisition and display, referred to as "video memory," and the second used as a scratchpad and communications for the data processing nodes, referred to as "scratchpad memory." Acquisition and display are programmed using the video library. The input look-up tables($LUTs$) and output $LUTs$ can be defined by the user in video library.

Chapter 3

Binary Morphological Filters

Mathematical morphology extracts information about the geometrical structure of an image object by transforming it through its interaction with another object, called the *structuring element*, which is of simpler shape and size than the original image object. Information about size, spatial distribution. shape connectivity, convexity, smoothness, and orientation can be obtained by transforming the image object using different structuring elements. In this chapter, we discuss two-valued Euclidean images, which are subsets of the Euclidean plane. Most of the notations, definitions and properties are adopted from [2], [3] and [4].

3.1 Fundamental Operations

Given an image (subset) A in the Euclidean plane R^2, the *translation* of A by the point x in R^2 is defined by [2]

$$A + x = \{a + x : a \in A\},$$

where the plus sign inside the set notation refers to vector addition. The following example illustrates the definition of translation arithmetically:

Example:

$A = \{(0,0),(1,0),(0,1),(1,1),(2,2)\}$,

$x = (3,1)$,

$A + x = \{(3,1),(4,1),(3,2),(4,2),(5,3)\}$.

0	0	1
1	1	0
1	1	0

A

0	0	0	0	0	1
0	0	0	1	1	0
0	0	0	1	1	0
0	0	0	0	0	0

$A + x$

The *Minkowski set addition* $A \oplus B$ of two sets A and B consists of all points c which can be expressed as an algebraic vector addition $c = a + b$, where the vectors a and b belong to the sets A and B, respectively.

Definition: The Minkowski addition of A and B is equal to the set union of all the translations $A + b$ of A when the vector b sweeps the set B [2]:

$$A \oplus B = \bigcup_{b \in B} A + b. \tag{3.1}$$

Example:

$A = \{(0,0),(1,0),(0,1),(1,1),(2,2)\}$,

$B = \{(0,0),(1,1)\}$,

$A \oplus B = \{(0,0),(1,0),(0,1),(1,1),(2,2),(2,1),(1,2),(3,3)\}$.

0	0	1
1	1	0
1	1	0

A

0	1
1	0

B

0	0	0	1
0	1	1	0
1	1	1	0
1	1	0	0

$A \oplus B$

The *Minkowski set subtraction* of B from A, denoted as $A \ominus B$, is the dual operation to the Minkowski set addition defined as follows:

$$A \ominus B = (A^c \oplus B)^c.$$

Definition: The Minkowski subtraction of A from B is equal to the set intersection of all the translations $A + b$ of A when the vector b sweeps the set B [2]:

$$A \ominus B = \bigcap_{b \in B} A + b. \tag{3.2}$$

Example:

$A = \{(0,0),(1,0),(0,1),(1,1),(2,2)\}$,

$B = \{(0,0),(1,1)\}$,

$A \ominus B = \{(1,1),(2,2)\}$.

0	0	1
1	1	0
1	1	0

A

0	1
1	0

B

0	0	1
0	1	0
0	0	0

$A \ominus B$

3.2 Morphological Operations

Morphological erosion and dilation are defined from a geometric point of view as set transformations that shrink or expand a set. Algebraically, however, they are actually Minkowski set subtraction and addition, respectively.

3.2.1 Erosion

Let the closed set A represent a binary image and the compact set B a structuring element. The complement A^c of A represents the image background. We denote by B^s the *symmetric* set of B with respect to origin; i.e., B^s is obtained by rotating B 180^0 on the plane so that $B^s = \{-b : b \in B\}$. Then, the *erosion* of A by B is defined geometrically as the set of those points z such that the translation $B + z$ is contained in the original image set A. Algebraically, the erosion of A by B is equal to the Minkowski set subtraction of B^s from A [2]:

$$A \ominus B^s = \{z : B_z \subseteq A\} = \bigcap_{b \in B} A + (-b). \tag{3.3}$$

Example:

1	1	1
1	♣1	1
1	1	1

A

0	0	1
0	♣1	0
0	0	0

B

0	0	0
0	♣1	0
1	0	0

B^s

0	1	1
0	♣1	1
0	0	0

$A \ominus B$

0	0	0
1	♣1	0
1	1	0

$A \ominus B^s$

where ♣ is the origin.

The example above shows the difference between Minkowski subtraction and erosion. If B is a symmetric structuring element, we can see that there is no difference between them.

3.2.2 Dilation

The dual operation of erosion with respect to complementation is the dilation. Dilating the object is equivalent to eroding its background. *Dilation* of A by B is defined geometrically as the set of all those points z such that the translation $B + z$ intersects A. Algebraically, the dilation of A by B is equal to the Minkowski sum of A and B^s [2]:

$$A \oplus B^s = \{z : B_z \cap A \neq \emptyset\} = \bigcup_{b \in B} A + (-b). \tag{3.4}$$

Example:

0	1	0
0	♣1	1
0	0	0

A

0	1	0
0	♣1	1
0	0	0

B

0	0	0
1	♣1	0
0	1	0

B^s

0	0	1	0	0
0	0	1	1	0
0	0	♣1	1	1
0	0	0	0	0
0	0	0	0	0

$A \oplus B$

1	1	0
1	♣1	1
0	1	1

$A \oplus B^s$

The example above shows the difference between Minkowski addition and dilation. If B is a symmetric structuring element, it is obvious that there is no difference between them.

3.2.3 Opening and Closing

Another pair of dual morphological transformations are the opening and closing. If we erode A by B and then dilate the eroded set $A \ominus B^s$ by B^s, we generally do not recover A, but rather a simplified and less detailed version of A. This new set is called the *opening*, A_B, of A by B. By duality, the *closing*, A^B, of A by B comes from dilating first and then eroding. Thus, the opening and closing are defined, respectively, as [2]

$$A_B = (A \ominus B^s) \oplus B^s \tag{3.5}$$

$$A^B = (A \oplus B^s) \ominus B^s. \tag{3.6}$$

If the structuring element B has a regular shape, both opening and closing smooth the contours of the object. The opening transformation suppresses the sharp capes and cuts the narrow isthmuses of the object, whereas the closing transformation fills up the thin gulfs and small holes. Both transformations are generally noninvertible.

3.3 Implementation and Complexity

To implement the basic morphological set operations on a conventional computer, one could simply represent the sets by binary functions, whose values are equal to one and zero at points of the object and its background, respectively. Taking the local minimum or the maximum inside a running window B will shrink or expand, respectively, the ones of this binary function [5].

Example:

0	0	1
1	1	0
1	1	0

A

0	1
1	0

B

Take the upper-right corner of the structuring element B as the kernel of the running window and replace the pixel value under the kernel with the minimum value of those pixels masked by one. After repeating this operation on the whole image, the erosion results.

0	0	1
0	1	0
0	0	0

$$A \ominus B$$

Since the parallel image processing work station is capable of processing the whole image in one operation, the parallel implementation of erosion and dilation can be accomplished. This is true from the fact that erosion and dilation of A by B are algebraically interpreted as the intersection and union, respectively, of all the translations $A + (-b)$ of A when b sweeps B.

The structuring element B is a compact set, which for discrete sets means that B contains a finite number of points. Thus, the number of required translations of A is finite. Fig 3.1 shows the parallel implementation [6]. We need two planes for the image A and the structuring element B and a third accumulation plane for the resulting transformed image. The image plane is shifted in parallel to the accumulation plane, and the amount of shifting is controlled by the points belonging to the structuring element B. The accumulator plane holds the parallel logical AND or OR of all the shifted versions of the image plane and after all the points of B have been spanned, it will contain the erosion or dilation, respectively, of the original image. The following example illustrates this implementation.

Example:

0	0	1
1	1	0
1	1	0

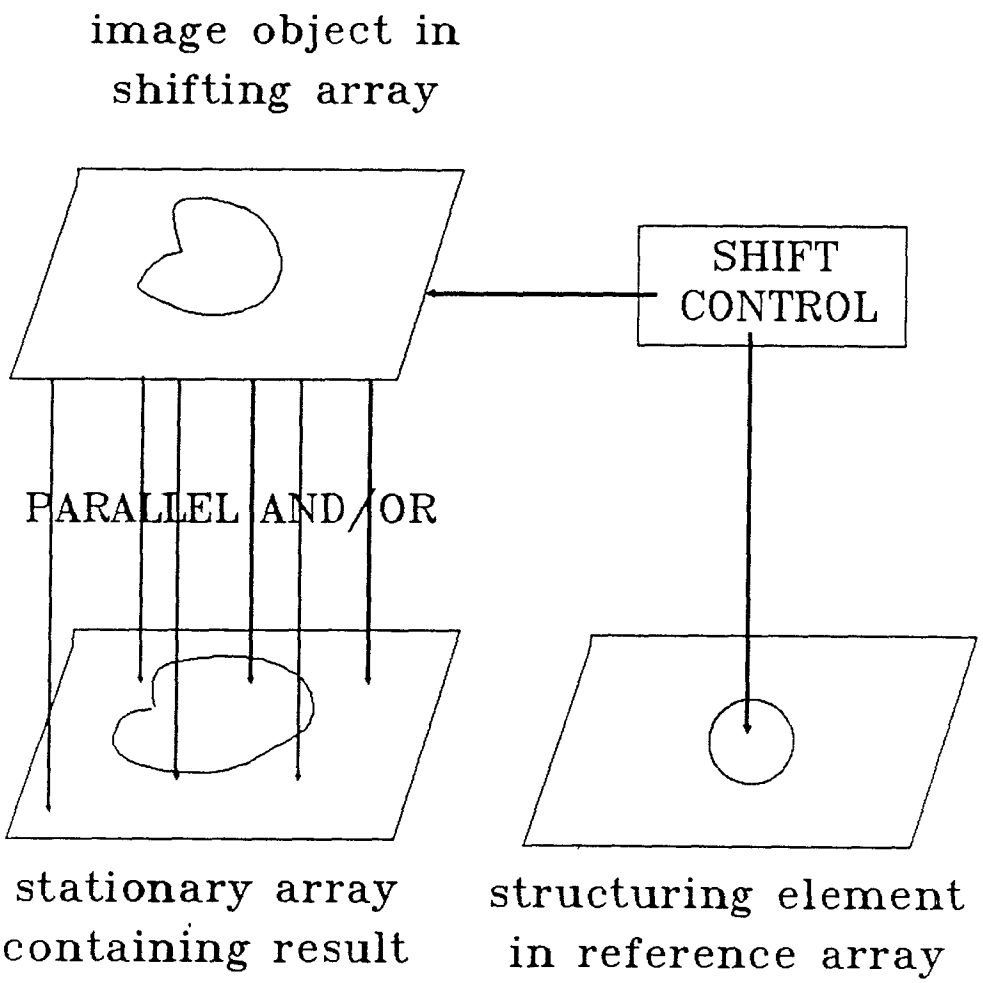


Figure 3.1: Parallel implementation of erosion and dilation

A

0	^{2}1
^{1}1	0

B

0	0	0	0
0	0	1	0
1	1	0	0
1	1	0	0

$C = shifted\ A\ according\ to\ ^1$

0	0	0	1
0	1	1	0
0	1	1	0
0	0	0	0

$D = shifted\ A\ according\ to\ ^2$

0	0	0	1
0	1	1	0
1	1	1	0
1	1	0	0

$A \oplus B = C$ OR D

From the discussion above, we see that it takes N^2 window operations to implement dilation or erosion of an $N \times N$ image on a conventional computer. For the parallel implementation, it takes at most M^2 image plane operations for an $M \times M$ structuring element. In general, M is much smaller than N. Thus, the computational complexity for these two implementations is obvious.

3.4 Androx Implementation

As described in the previous section, to implement a binary morphological operation using the second method, the only two operations required are *SHIFT* and *AND* (or *OR*).

To implement *SHIFT* operation in Androx, we can simply copy the original image to a specific position in video memory, then change the position of the origin of the image according to the position of the occurrence of 1's in the structuring element. In Androx, we always take the upper left corner of the image as the origin. Fig. 3.2 illustrates this operation. The area of the video memory to which the image is copied must be reset before performing the *SHIFT* operation to avoid the distortion of the resulting image. The next *SHIFT* operation can be done by simply changing the origin of the image which has been copied to the specific position in the video memory.

The *AND* and *OR* operations can be done by calling the AND or OR image processing library functions. Since calling these functions must specify the origin of the two images that are of interest, the *SHIFT* operation is actually implied in these functions. The result of the *AND* (or *OR*) is output to the position where the original image locates. So this position will contain all partial results until the final result is achieved.

Example:

To *AND* an image at (x1, y1) to another image at (x2, y2) in video mem-

ory, we can call the AND function in the Androx image processing library as following:

isp_event(AND | BRD(board_number), num_bits, x1, y1, xlen, ylen, x2, y2,
xout, yout).

board_number is number of Androx. (In our system, we have only one board. Therefor, the board_number is 0.)

num_bits specifies the pixel size.

x1 and y1 are the origin of the first image of interest. (For the case shown in Fig. 3.2, x1=0 and x2=0.)

xlen and ylen are the length and height, respectively, of the image. (In our case, xlen=ylen=512.)

x2 and y2 are the origin of the second image of interest. (For the case shown in Fig. 3.2, x2=512, y2=1024.)

xout and yout are the origin where the output result is to be stored. (For the case in Fig. 3.2, xout=0, yout=0.)

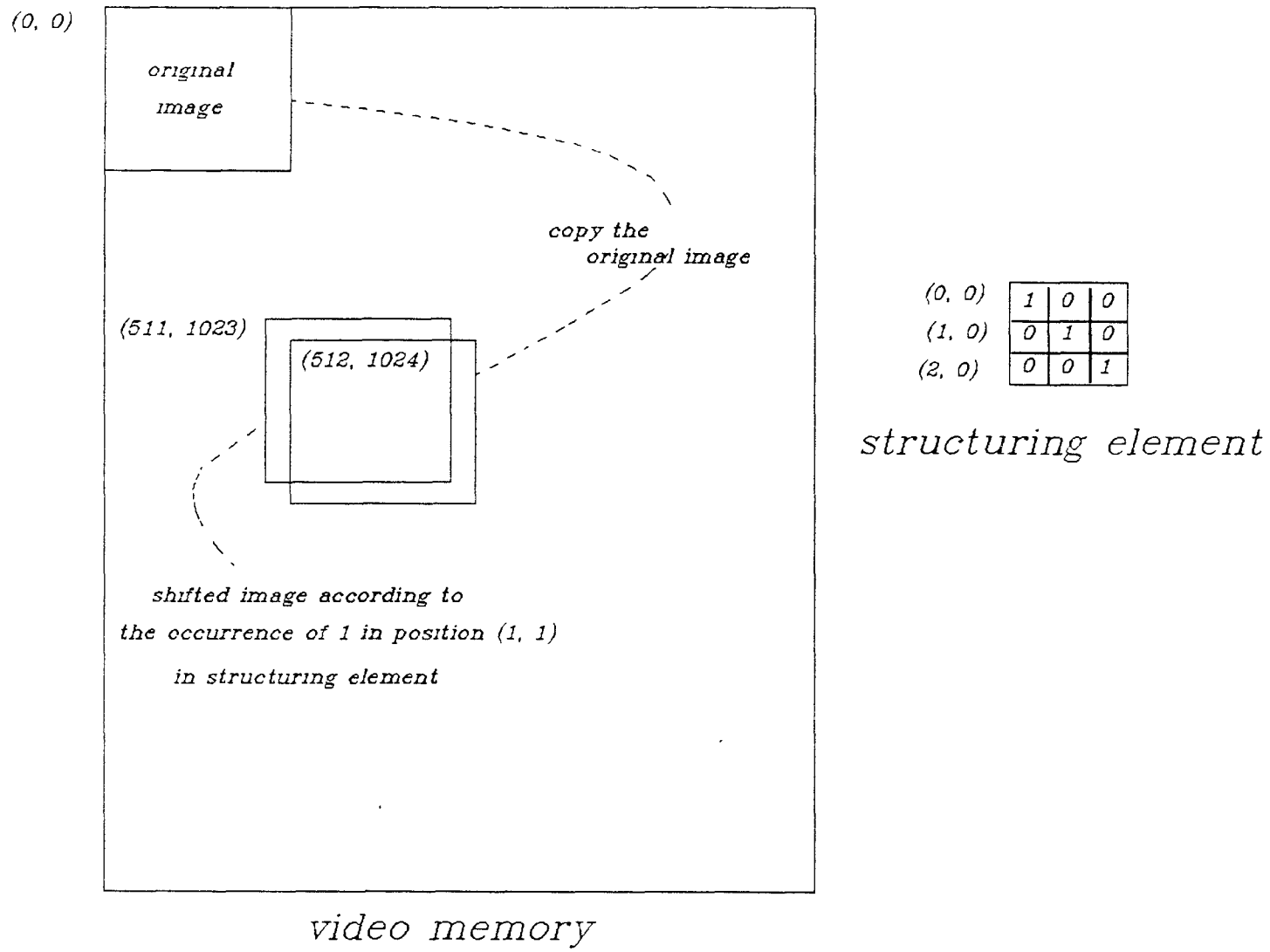
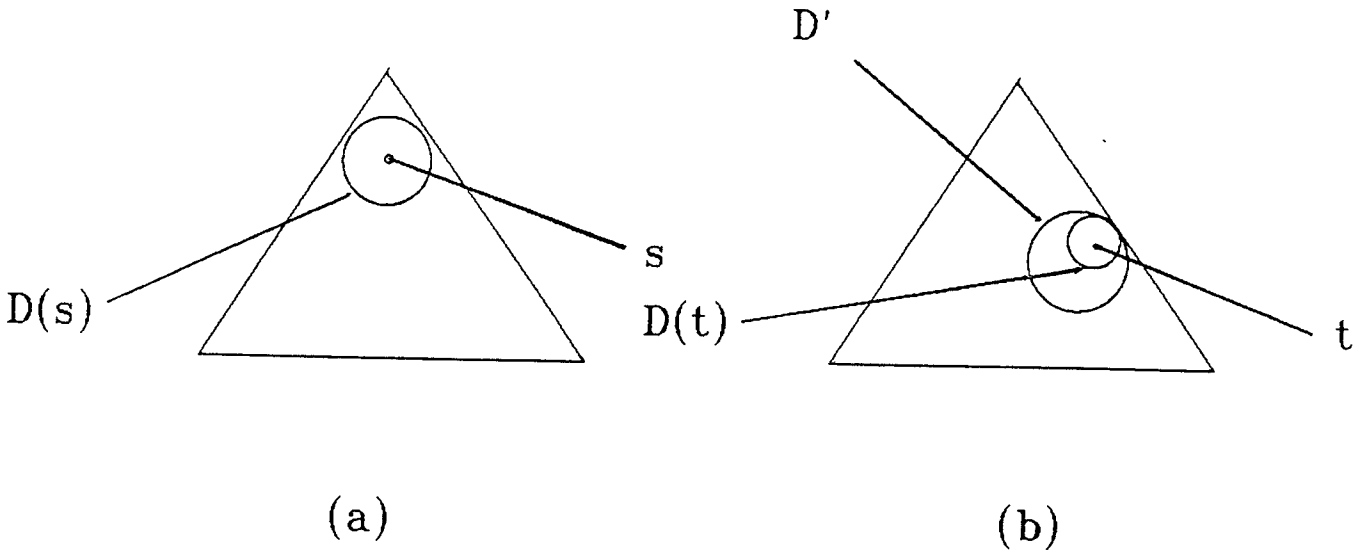


Figure 3.2: Implementing *SHIFT* Operation in Androx

Chapter 4

Thinning

An important approach for representing the structural shape of a plane region is to reduce it to a graph. This is often accomplished by obtaining the skeleton of the region via a thinning algorithm. Thinning algorithms play a central role in a broad range of problems in image processing, ranging from automated inspection of printed circuit board to counting of asbestos fibers in air filter [2]. In general, thinning process is defined as the successive removal of outer layers of pixels from an object while retaining any pixels whose removal would alter the connectivity. Many thinning algorithms have been proposed to date. Two algorithms, based on morphological set transformation, are discussed in this chapter. Experimental results are used to compare these two thinning algorithms. The correctedness of both algorithms, such as convergence, one-pixel thickness, and connectedness, are discussed in [7], and [8].



4.1 Medial Axis

The medial axis (skeleton) of a continuous image object X, viewed as a subset of R^2, is defined as the set of the centers of the maximal disks inscribable inside X. A disk is *maximal* [8] if it is not properly contained in any other disk totally included in X. Hence a maximal disk must touch the boundary of the object X at least at two different points. For example, consider the triangles in Fig. 4.1. Note that, the point s lies in the medial axis, since $D(s)$ is a maximal disk (Fig. 4.1(a)), the point t does not lie in the medial axis, since $D(t)$ is not a maximal disk D'(Fig. 4.1(b)).

4.2 Hit/Miss and Thin Operator

An important morphological operation used in thinning algorithm is the hit/miss transform [2]. Let structuring element B be composed of two subsets B^1 and B^2; then the hit/miss transform of image X by B is defined as the set of all points where B_x^1 is included in X and B_x^2 is included in X^c. The set X^c is the complement of X and B_x^i, $i = 1, 2$, denotes the translation of B^i by x. This hit/miss operation is expressed as [8]

$$\begin{aligned} X \otimes B &= \{x : B_x^1 \subset X; B_x^2 \subset X^c\} \\ &= (X \ominus B^{s1}) \cap (X^c \ominus B^{s2}) \end{aligned} \tag{4.1}$$

where $X \ominus B^{s1}$ is the erosion of X by B^1. If B^2 is chosen as the window complement of B^1, (4.1) can be rewritten as

$$X \otimes B = (X \ominus B^{s1}) \cap (X^c \ominus (W \cap (B^s)^c)) \tag{4.2}$$

where W is the window with finite support.

If we form a window with the same size as the structuring element by taking a pixel in an image as the kernel, the *hit/miss* transform can be understand as follows: A pixel is said to *hit* the structuring element if the window formed is exactly the same as the structuring element. Otherwise, it *misses*. The hit/miss operation can be used to detect the occurrence of the *exact* pattern B^1 in the image X. The following example shows how a pattern having the shape of the structuring element in an image is detected by the hit/miss transform.

Example:

0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	1	1	0	0	1	0	0
0	1	1	1	1	1	1	1	1	0
0	0	1	1	1	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

X

1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1
1	1	0	0	0	1	1	0	1	1
1	0	0	0	0	0	0	0	0	1
1	1	0	0	0	1	1	0	1	1
1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

X^c

0	1	0
1	♣1	1
0	1	0

B

1	0	1
0	♣0	0
1	0	1

$W \cap B^c$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	1	1	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$C = X \ominus B^s$$

1	1	0	1	0	1	1	1	1	1
1	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	1
1	1	0	1	0	1	1	1	1	1

$$D = X^c \ominus (W \cap (B^s)^c)$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$X \otimes B = C \cap D$$

Before defining the thinning transform, the set difference must be introduced first. The set difference of X and Y, denoted as X/Y, is defined as [8]

$$X/Y = (X^c \cup Y)^c.$$

With the definition of hit/miss transform, the *thinning* of X by B is defined as [8]

$$X \bigcirc B = X/(X \otimes B). \tag{4.3}$$

The thinning operation is actually a search-and-delete process [6]. The operation $X \otimes B$ locates all occurrence of the structuring element B in X and the operation / removes from X those points which have been located. The skeleton of an image can be acquired by repeating this operation until no further pixel can be removed.

4.3 Thinning Algorithms

4.3.1 The First Algorithm

The first algorithm described here represents a general class of thinning algorithms that has been proposed by numerous researches in the past. It consists of two cycles. The first cycle is an iterative process in which each iteration consists of four passes. Each pass uses a 3×3 structuring element to remove border points from a given direction. In other words, points that satisfy the hit/miss transform of the structuring element are removed.

To complete one iteration, a sequence of four structuring element, one for each pass, are applied to remove border points from all four directions, that is, the northwest, northeast, southeast, and southwest. The four structuring element are shown in Fig. 4.2 as $\mathbf{D} = \{D^1, D^2, D^3, D^4\}$. The second cycle is another iterative process designed to remove north, east, south, and west border points as well as unnecessary points at junctions (i.e. intersections of two lines). The four structuring elements, $\mathbf{E} = \{E^1, E^2, E^3, E^4\}$, used for this purpose are shown in Fig. 4.2.

Let X be a connected component and I be the resulting skeleton from the thinning algorithm. The iteration process of the first and second cycles are indexed by m and n, respectively. After a finite number of iterations, depending on the maximum thickness of X, X converges to the skeleton I, which is a stick-like figure preserving the connectivity of X.

This algorithm is precisely specified by the transformations ψ_1 and ψ_2,

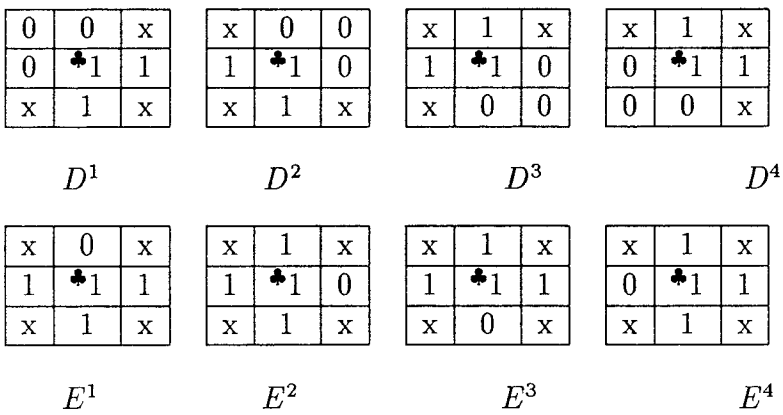


Figure 4.2: Structuring Elements for Thinning Algorithm

representing the first and second cycles, respectively. The first transformation is given by

$$\begin{aligned} \psi_1\{X, \mathbf{D}, m\} &= \{X \bigcirc \mathbf{D}\}_m \\ &= (\cdots(((((X \bigcirc D^1) \bigcirc D^2) \bigcirc D^3) \bigcirc D^4) \\ &\quad \bigcirc D^1) \cdots) \end{aligned} \tag{4.4}$$

where $X \bigcirc \mathbf{D}$ is the thinning of X by the four structuring elements in $\mathbf{D}$ and $\{X \bigcirc \mathbf{D}\}_m$ is the process repeated m times. Assume for the time being that $\{X \bigcirc \mathbf{D}\}_m$ converges in M iterations; the result becomes

$$X' = \psi_1\{X, \mathbf{D}, M\}. \tag{4.5}$$

The second transformation is given by

$$\psi_2\{X', \mathbf{E}, n\} = \{X' \bigcirc \mathbf{E}\}_n \tag{4.6}$$

and it converges to I in N iterations.

4.3.2 The Second Algorithm

The second algorithm [7] described here uses the same sets of structuring elements, **D** and **E**, as in the first algorithm. However, the order of applying these structuring elements is different. In fact, three structuring elements are applied simultaneously in each pass for this algorithm. Besides, this algorithm consists of only one cycle for each iteration.

The transformation ψ of the second algorithm is defined as [7]

$$\psi(S, \mathbf{D}, \mathbf{E}, m) = \{(S \bigcirc D^i) \cap (S \bigcirc D^{i+1} \cap (S \bigcirc E^i)\}_m \qquad (4.7)$$

where $i = 1, 2, 3$, or 4 is the index of the structuring elements and a different i must be chosen for each pass. Fig. 4.3 shows how this algorithm works. There are a total of four passes in each iteration and the iteration index is defined by m. The structuring elements D^i, D^{i+1}, and E^i are applied in parallel at each pass.

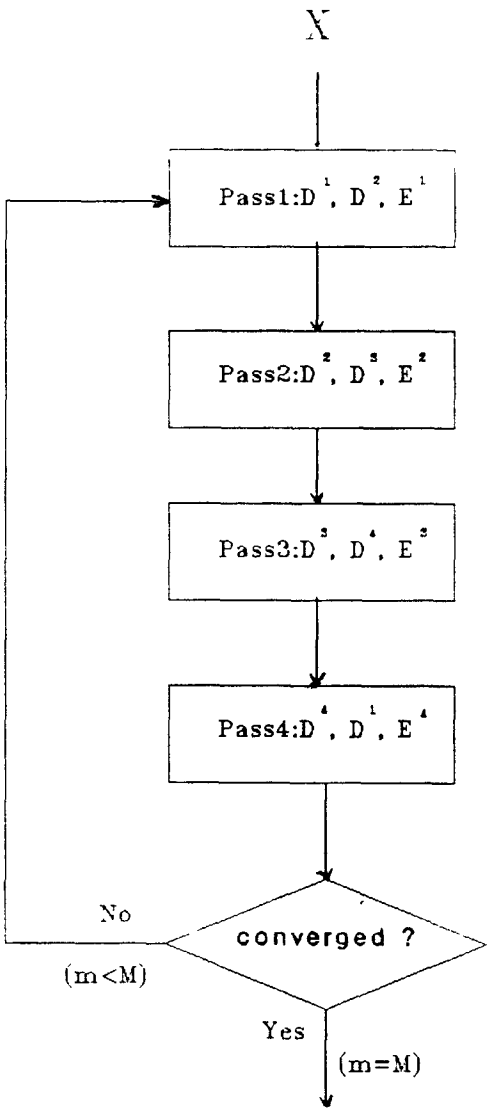


Figure 4.3: Second Algorithm [7]

4.4 Androx Implementation

To implement th hit/miss transform in Androx, we do not have to go from taking the erosion of the image (i.e., Equation (4.2)). For a specific structuring element, we can form two running windows which are of the same size as the structuring element. For the first window, we mask those relative positions where the 1's are located in the structuring element. Then count the number of pixels with value 255 under the position which has been masked. If the number counted is the same as the number masked, then replace the value under the kernel with value 0. After repeating this running window on the whole image, the output is placed in a buffer for later use. For the second window, we mask those positions where the 0's are located in the structuring element. Then count the number of pixels with value 0 under the masked positions. If the number counted is the same as the number masked, then replace the value under the kernel with value 0. The output resulting from the second window is ORed with the output of the first window. The result of this OR operation is actually the result of the thining transform rather than the hit/miss transform. This is true from the fact that we have replace those pixels which hit the structuring element with value 0, implying that these pixels have been removed. Fig. 4.4 illustrates the operations described above.

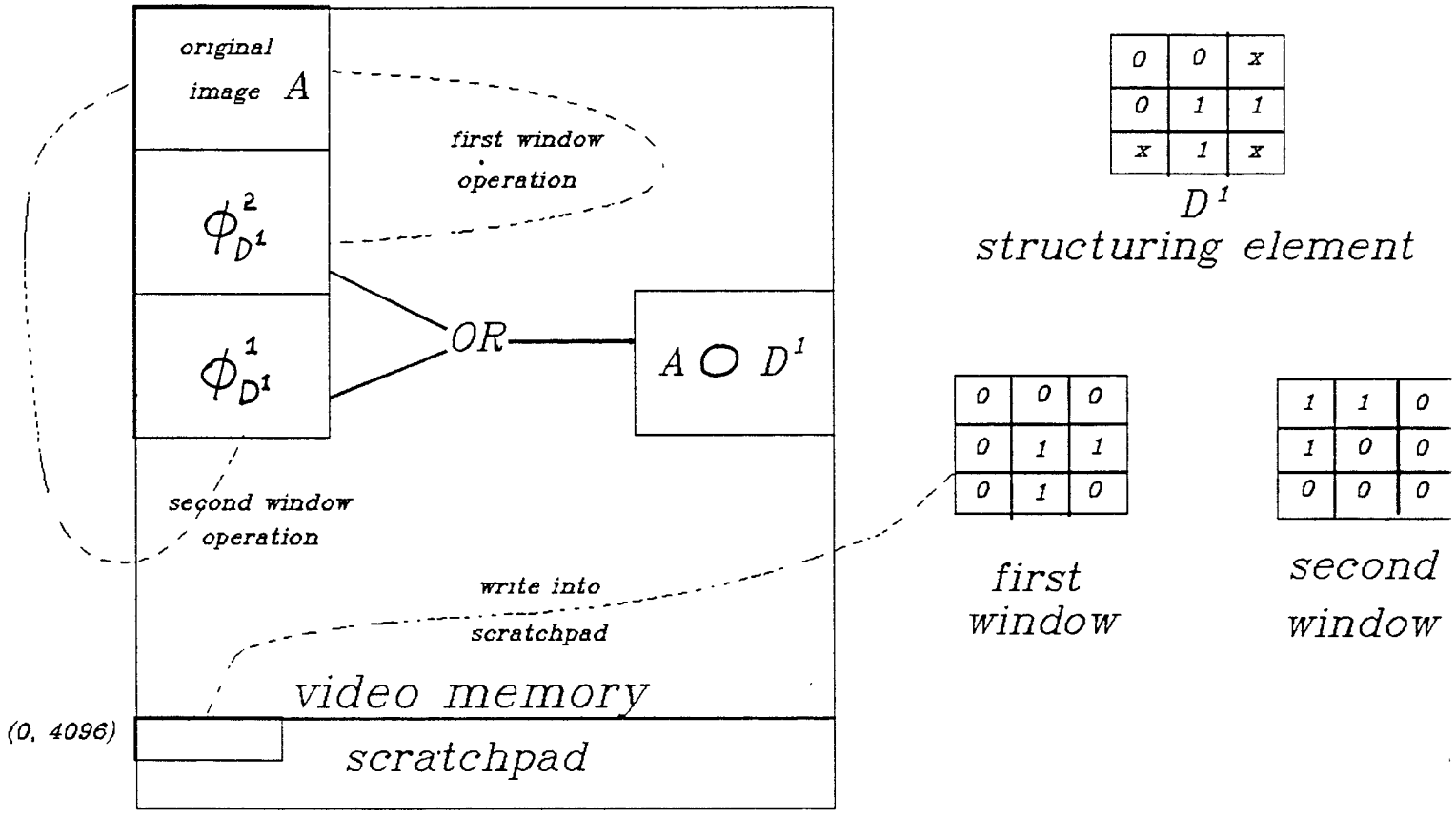


Figure 4.4: Implementation of Thining Transform on Androx

Let ϕ_B^1 denotes the result of the first window operation on image A and ϕ_B^2 denotes the result of the second window operation on image A, then the thining of A by B can be expressed by:

$$A \bigcirc B = \phi_B^1 \text{ OR } \phi_B^2.$$

The window operation described above can be done by calling the REPLACE_EQUAL_MASK function in the image processing library of Androx. This function replaces the center pixel under a two-dimensional kernel if a specified number of the pixels under the masked element are equal to a specified threshold. The usage of REPLACE_EQUAL_MASK is as follows:

isp_event(REPLACE_EQUAL_MASK | BRD(0), num_bits, x, y, xlen, ylen, kx, ky, xout, yout, scrx, scry, scr_xlen, scr_ylen, constant, limit, threshold)

kx and ky are the horizontal and vertical dimension, respectively, of the kernel. (For the case shown in Fig. 4.4, $kx = ky = 3$.)

$scrx$ and $scry$ are the coordinates of the kernel storage area. Kernel coefficients must be written to the scratchpad before the function is called. (For the first window operation shown in Fig. 4.4, $scrx = 0$ and $scry = 4096$.)

scr_xlen is the number of bytes per row in the kernel storage area. (For the case shown in Fig. 4.4, $scr\_xlen = 18$.)

scr_ylen is the number of rows in the kernel storage area. (For the case shown in Fig. 4.4, $scr\_ylen = 1$.)

constant is the value with which to replace the center pixel under the kernel if conditions for replacement are met. (In our case, *constant* = 0, which imply the removal of the pixels that hit the mask.)

limit is the number of pixels under the masked elements of the kernel which must meet the threshold requirements for replacement to be made. (For the first window operation in Fig. 4.4, *limit* = 3.)

threshold is the pixel value against which the pixels under the masked elements of the kernel are checked. (Since we are working on binary images using the 8-bit memory plane, the pixel value 0 is represented in a 8-bit memory plane by value 255. For the first window operation in Fig.4.4, *threshold* = 255. For the second window operation, *threshold* = 255.)

Having the implemented the thinning transform, both thinning algorithms discussed in previous section can be easily implemented. The first algorithm can be done by simply using Equations (4.4), (4.5), and (4.6). To implement each pass in Fig. 4.3 of the second algorithm, one can AND the partial results which result from thining the image A by the three structuring elements in each pass. Let φ_1 denote the partial result of pass1 as indicated in Fig. 4.3, φ_1 can be implemented by

$$\varphi_1 = (A \bigcirc D^1) \text{ AND } (A \bigcirc D^2) \text{ AND } (A \bigcirc E^1).$$

The second algorithm is then implemented by repeating the four passes as shown in Fig. 4.3 until no further pixel can be removed.

4.5 Discussion

From the previous section, we see that a set of structuring elements is required to thin an image X. The only difference between the two algorithms is the order of applying the same structuring elements. As we can see in Fig. 4.5, these two algorithms result in different skeletons for the same image. Fig. 4.5(b) shows the skeleton obtained by the first algorithm. It differs far from the medial axis which is what we expected. This is due to the fact that convex corners are removed by **D** first in the first cycle and then concave corners are removed by **E** in the second cycle when no further convex corners can be removed. Thus, convex corners are removed faster than concave corners.

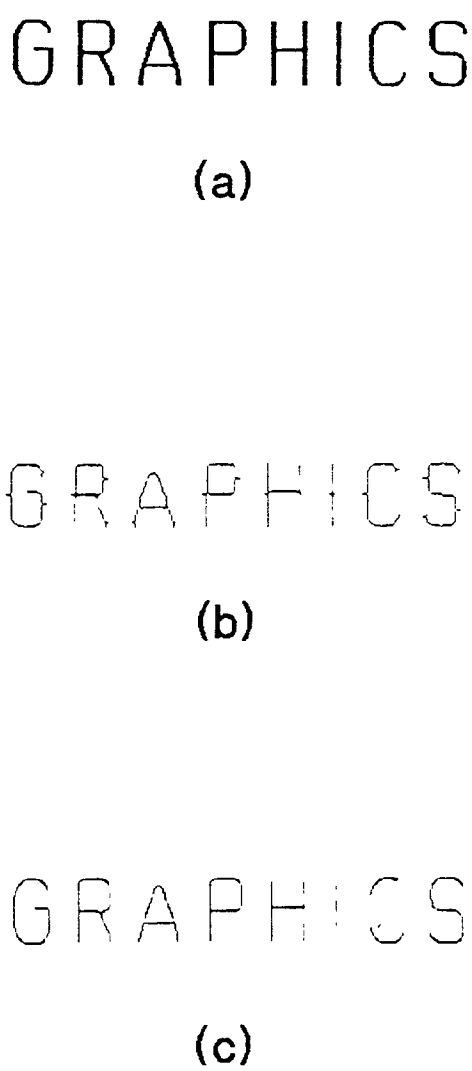


Figure 4.5: Skeletons from the first and second algorithm

Chapter 5

Gray-Scale Image Filters

5.1 Umbra and Top Surface

Two very useful devices, *top surface* and *umbra* transform, for the relating gray-scale morphology must be introduced before defining gray-scale erosion and dilation.

Let A be a subset in Euclidean plane R^2. The *umbra* of A, denoted $U[A]$, is the set of all points that lie beneath some point of A and defined by [4]

$$U[A] = \{(x, y) : \textit{there exists a } y' \textit{ with } (x, y') \in A \textit{ and } y \leq y'\}. \quad (5.1)$$

The *top surface* of A, denoted $T[A]$, is the set of all points (x, y_x) such that x is in the domain of A and

$$y_x = max\{y : (x, y) \in A\}. \quad (5.2)$$

Note that the surface transform is actually the inverse mapping of umbra transform(i.e $U^{-1} = T$). The following example illustrates this property.

Example:

6	0	0	0	0	0	0	0	0	0
5	0	0	1	0	0	0	0	0	0
4	0	1	0	1	0	0	1	0	0
3	1	0	0	0	1	1	0	1	0
2	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
	0	1	2	3	4	5	6	7	8

$A = T[U[A]]$

6	0	0	0	0	0	0	0	0	0
5	0	0	1	0	0	0	0	0	0
4	0	1	1	1	0	0	1	0	0
3	1	1	1	1	1	1	1	1	0
2	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1
	0	1	2	3	4	5	6	7	8

$U[A]$

5.2 Dilation, Erosion, Opening and Closing

The gray scale dilation of A by B, denoted $A \oplus_g B$, is defined by

$$A \oplus_g B = T[U[A] \oplus_b U[B]]. \tag{5.3}$$

where $\oplus_b$ is the binary dilating operator(i.e., $C \oplus_b D = C \oplus D^s$). We use the notation $\oplus_b$ only for convenience.

Example:

6	0	0	0	0	0	0
5	0	0	1	0	0	0
4	0	1	0	1	0	0
3	1	0	0	0	1	0
2	0	0	0	0	0	0
1	0	0	0	0	0	0
0	0	0	0	0	0	0
	0	1	2	3	4	5

A

6	0	0	0	0	0	0
5	0	0	1	0	0	0
4	0	1	1	1	0	0
3	1	1	1	1	1	0
2	1	1	1	1	1	0
1	1	1	1	1	1	0
0	1	1	1	1	1	0
	0	1	2	3	4	5

$U[A]$

1	0	1
0	1	0
	0	1

B

1	0	1
0	1	1
	0	1

$U[B]$

6	0	0	0	1	0	0
5	0	0	1	1	1	0
4	0	1	1	1	1	1
3	1	1	1	1	1	1
2	1	1	1	1	1	1
1	1	1	1	1	1	1
0	1	1	1	1	1	1
	0	1	2	3	4	5

$U[A] \oplus_b U[B]$

6	0	0	0	1	0	0
5	0	0	1	0	1	0
4	0	1	0	0	0	1
3	1	0	0	0	0	0
2	0	0	0	0	0	0
1	0	0	0	0	0	0
0	0	0	0	0	0	0
	0	1	2	3	4	5

$A \oplus_g B = T[U[A] \oplus_b U[B]]$

The dilation operation can be equivalently written as the following property.

Property: Let $f : F \rightarrow E$ and $k : K \rightarrow E$, then $f \oplus_g k : F \oplus_g K \rightarrow E$ can be computed by

$$(f \oplus_g k)(x) = max\{f(x-z) + k(z)\}, \forall z \in K \ \ and \ \ x - z \in F. \qquad (5.4)$$

The gray scale erosion of A by B, denoted $A \ominus_g B$, is defined by

$$A \ominus_g B = T[U[A] \ominus_b U[B]]. \tag{5.5}$$

Example:

6	0	0	0	0	0	0
5	0	0	1	0	0	0
4	0	1	0	1	0	0
3	1	0	0	0	1	0
2	0	0	0	0	0	0
1	0	0	0	0	0	0
0	0	0	0	0	0	0
	0	1	2	3	4	5

A

6	0	0	0	0	0	0
5	0	0	1	0	0	0
4	0	1	1	1	0	0
3	1	1	1	1	1	0
2	1	1	1	1	1	0
1	1	1	1	1	1	0
0	1	1	1	1	1	0
	0	1	2	3	4	5

$U[A]$

1	1	0
0	0	1
	0	1

B

1	1	0
0	1	1
	0	1

$U[B]$

6	0	0	0	0	0	0
5	0	0	0	0	0	0
4	0	0	1	1	0	0
3	0	1	1	1	0	0
2	1	1	1	1	0	0
1	1	1	1	1	0	0
0	1	1	1	1	0	0
	0	1	2	3	4	5

$U[A] \ominus_b U[B]$

6	0	0	0	0	0	0
5	0	0	0	0	0	0
4	0	0	1	1	0	0
3	0	1	0	0	0	0
2	1	0	0	0	0	0
1	0	0	0	0	0	0
0	0	0	0	0	0	0
	0	1	2	3	4	5

$$A \ominus_g B = T[U[A] \ominus_b U[B]]$$

The erosion operation can be equivalently written as the following property.

Property: Let $f : F \rightarrow E$ and $k : K \rightarrow E$, then $f \ominus_g k : F \ominus_g K \rightarrow E$ can be computed by

$$(f \ominus_g k)(x) = min\{f(x+z) - k(z)\}, \forall z \in K \ and \ x+z \in F. \tag{5.6}$$

The gray scale opening and closing are defined similarly to the binary opening and closing in chapter 3. The gray scale opening of image A by B, denoted $OPEN(A, B)$, is defined by

$$OPEN(A, B) = (A \ominus_g B) \oplus_g B. \tag{5.7}$$

The gray scale closing of image A by B, denoted $CLOSE(A, B)$, is defined by

$$CLOSE(A, B) = (A \oplus_g B) \ominus_g B. \tag{5.8}$$

5.3 Implementation and Complexity

Two methods for implementing both dilation and erosion are discussed here. The first method [4] uses the threshold decomposition property, which are implied in Equations (5.3) and (5.5). The second method is directly derived from the properties defined in Equations (5.4) and (5.6).

5.3.1 The First Method

According to the threshold decomposition property, a gray scale image can be converted into a stack of binary images by slice and slice complement technique [4] Slice and slice complement are defined as:

Definition: Let $F \subseteq E^{N-1}$ and $f : F \rightarrow E$. The slice of f, denoted by $S[f]$, $S[f] \subseteq F \times E$, is defined by

$$\begin{aligned} S[f_m]_{(x,y)} &= 1 \ if \ y = m \ and \ f(x) \geq y \\ &= 0 \ otherwise \end{aligned}$$

where $x \in E^{N-1}, y \in E$.

Definition: Let $F \subseteq E^{N-1}$ and $f : F \rightarrow E$. The slice complement of f, denoted by $\bar{S}[f]$, $\bar{S}[f] \subseteq F \times E$, is defined by

$$\begin{aligned} \bar{S}[f_m]_{(x,y)} &= 1 \ if \ y = m \ and \ f(x) \geq y \\ &= 0 \ if \ y = m \ and \ f(x) < y \\ &= 1 \ y \neq m \end{aligned}$$

where $x \in E^{N-1}, y \in E$. For a disecrete image f with L levels (0, 1, ..., $L-1$), the umbra of the image can be decomposed into L slices, and

$$\begin{aligned} U[f] &= S[f_0] \cup s[f_1] \cup \cdots \cup s[f_{L-1}] \\ &= \bigcup_{m=0}^{L-1} S[f_m] \\ &= \bar{S}[f_0] \cap \bar{S}[f_1] \cap \cdots \cap \bar{S}[f_{L-1}] \\ &= \bigcap_{m=0}^{L-1} \bar{S}[f_m] \\ &= \bigcap_{m=0}^{L-1} f_{mb}. \end{aligned}$$

where f_{mb} is equivalent to $\bar{S}[f_m]$.

Example:

3	0	0	0	0
2	0	0	1	0
1	0	1	1	1
0	1	1	1	1
	0	1	2	3

$U[f]$

3	0	0	0	0
2	0	0	0	0
1	0	1	1	1
0	0	0	0	0
	0	1	2	3

$S[f_1]$

3	0	0	0	0
2	0	0	1	0
1	0	0	0	0
0	0	0	0	0
	0	1	2	3

$S[f_2]$

3	1	1	1	1
2	1	1	1	1
1	0	1	1	1
0	1	1	1	1
	0	1	2	3

$\bar{S}[f_1]$

3	1	1	1	1
2	0	0	1	0
1	1	1	1	1
0	1	1	1	1
	0	1	2	3

$$\bar{S}[f_2]$$

Assume P is the highest gray scale of the image, and Q is the highest gray scale of the structuring element. By Equation (5.3), the gray scale dilation operation can be expanded into binary dilations as follows [4]:

$$\begin{aligned}
(f \oplus_g k)(x,y) \;=\; & [f_{Pb} \oplus_b k_{Qb}] + \\
& [(f_{Pb} \oplus_b k_{(Q-1)b}) \cup (f_{(P-1)b} \oplus_b k_{Qb})] + \\
& \vdots \\
& +[(f_{Pb} \oplus_b k_{0b}) \cup (f_{(P-1)b} \oplus_b k_{1b}) \cup \cdots \cup (f_{(P-Q)b} \oplus_b k_{Qb})] \\
& +[(f_{(P-1)b} \oplus_b k_{0b}) \cup (f_{(P-2)b} \oplus_b k_{1b}) \cup \cdots \cup (f_{(P-Q-1)b} \oplus_b k_{Qb})] \\
& \vdots \\
& +[(f_{(Q+1)b} \oplus_b k_{0b}) \cup (f_{Qb} \oplus_b k_{1b}) \cup \cdots \cup (f_{1b} \oplus_b k_{Qb})] \\
& +[(f_{Qb} \oplus_b k_{0b}) \cup (f_{(Q-1)b} \oplus_b k_{1b}) \cup \cdots \cup (f_{0b} \oplus_b k_{Qb})] \\
& +[(f_{(Q-1)b} \oplus_b k_{0b}) \cup (f_{(Q-2)b} \oplus_b k_{1b}) \cup \cdots \cup (f_{0b} \oplus_b k_{(Q-1)b})] \\
& \vdots \\
& +[(f_{1b} \oplus_b k_{0b}) \cup (f_{0b} \oplus_b k_{1b})] \\
& +[(f_{0b} \oplus_b k_{0b})] \qquad (5.9)
\end{aligned}$$

By Equation (5.5), the gray scale erosion operation can be expanded into binary erosion as follows:

$$
\begin{aligned}
(f \ominus_g k)(x,y) \;=\; & [f_{1b} \ominus_b k_{Qb}] + \\
& [(f_{2b} \ominus_b k_{Qb}) \cap (f_{1b} \ominus_b k_{(Q-1)b})] + \\
& [(f_{3b} \ominus_b k_{Qb}) \cap (f_{2b} \ominus_b k_{(Q-1)b}) \cap (f_{1b} \ominus_b k_{(Q-2)b})] + \\
& \vdots \\
& +[(f_{Qb} \ominus_b k_{Qb}) \cap (f_{(Q-1)b} \ominus_b k_{(Q-1)b}) \cap \cdots \cap (f_{1b} \ominus_b k_{1b})] \\
& +[(f_{(Q+1)b} \ominus_b k_{Qb}) \cap (f_{Qb} \ominus_b k_{(Q-1)b}) \cap \cdots \cap (f_{2b} \ominus_b k_{1b})] \\
& \vdots \\
& [(f_{Pb} \ominus_b k_{Qb}) \cap (f_{(P-1)b} \ominus_b k_{(Q-1)b}) \cap \cdots \\
& \cdots \cap (f_{(P-Q+1)b} \ominus_b k_{1b})]
\end{aligned} \tag{5.10}
$$

5.3.2 The Second Method

The second method, derived directly from Equations (5.4) and (5.6), is similar to the local maximum and minimum implementation discussed in Chapter 3. The method for implementing the dilation is as follows:

> Rotate the structuring element 180^0 and take the upper right corner as the kernel of the window. Add the value of the window and the corresponding value of the image. Replace the value under the kernel with the maximum value of the result of the addition.

Example:

0	1	2	3	1
	0	1	2	3

f

0	1	2
	0	1

k

0	2	3	4	5
	0	1	2	3

$f \oplus_g k$

The method for implementing the erosion is as follows:

> Take the lower left corner as the kernel of the window. Subtract the value of the image by the corresponding value of the window. Replace the value under the kernel with the minimum value of the result of the subtraction.

Example:

0	1	2	3	1
	0	1	2	3

f

0	1	2
	0	1

k

0	0	1	−1	0
	0	1	2	3

$f \ominus_g k$

From the discussion above, we see that it takes $P \times Q \times M^2$ image plane operations for a $M \times M$ structuring element in the threshold decomposing method(i.e., the first method). However, the second method takes only $N \times N$ window operations. Taking a 512×512 image with $P = 256$ eroded by a 3×3 structuring element as an example, we see that it takes $256 \times 256 \times 3^2 = 589,824$ operations on the whole image plane for the first method, but it takes only $512 \times 512 = 262,144$ window operations for the second method. Moreover, we see that the operations for the first method increases with square of the dimension of the structuring element. The operation for the second method is fixed. Therefore, the computational complexity is obviously more for the first method. The reason that the first method is not suitable is from the fact that most existing general purpose computer are byte addressable. When implementing the threshold decomposing method, the binary bit operation must be done by byte operation in the Androx workstation, and thus it takes more time. The threshold decomposition method is well suitable for implementing the gray scale dilation or erosion by hardware.

5.4 Androx Implementation

The implementation of the second method for the dilation differs from the implementation of the erosion in two ways. First, the structuring element must be rotated for the dilation, but not for the erosion. Second, the dilation takes the maximum value of window sums, but the erosion take the minimum value of the window differences.

The required rotation of the structuring element for erosion can be done when reading the input structuring element. the following C code illustrates this implementation.

```
for (j=0; j<t; j++)
    for (i=0; i<s; i++)
        rotated[s-1-i][t-1-j] = original[i][j];
```

s and t are the horizontal and vertical size of the structuring element.

To find a maximum and a minimum value within a window, we can simply call the MAX and MIN function in the Androx image processing library as the following example:

Example:

To find the maximum pixel value of a *xlen* × *ylen* window at (x, y), we can use

isp_event(MAX | BRD(0), num_bits, x, y, xlen, ylen, xout, yout).

The maximum pixel value is written to the position (xout, yout).

The implementation of the dilation using the second method in Androx is illustrated in Fig. 5.1.

One important notion must be taken when taking the addition of two windows. As we can see in previous section, the maximum value of the the dilation is $P + Q$. In general, for an 8-bit pixel image, $P + Q$ is greater than the maximum value of the 8-bit pixel value. So the better way to output the addition is to output the result in 16 bits. This can be controlled by the *num_bits* in the Androx image processing library.

Example:

To add a $xlen \times ylen$ window at (x1, y1) in an image to a rotated structuring element at (x2, y2) in scratchpad, we can call the ADD function in the Androx library as follows:

isp_event(ADD | BRD(0), num_bits, x1, y2, xlen, ylen, x2, y2, xout, yout)

num_bits:

P8_8: 8 bits in, 8 bits out

P8_16: 8 bits in, 16 bits out.

The resulting window at (xout, yout) is a window with 16-bit pixel value if we substitute P8_16 into num_bits.

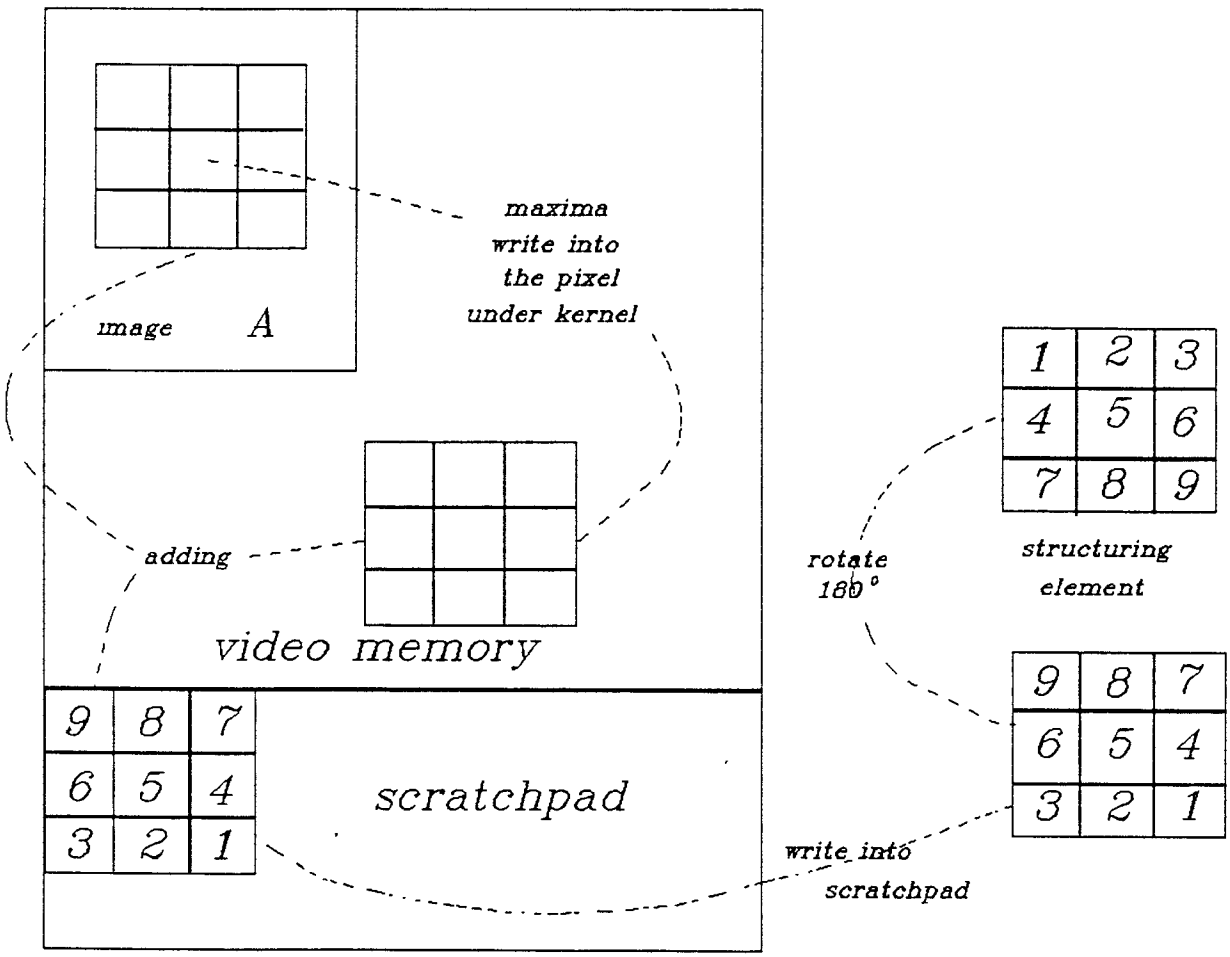


Figure 5.1: Androx Implementation of Second Method

Chapter 6

Conclusion

With the availability of the parallel image processing work station, fast binary morphological set erosions and dilations can be implemented simply using only parallel shifting and logical AND/OR operation on the image. Using the erosion, a very useful transform in pattern recognition, the hit/miss transform can be implemented. With the hit/miss transform defined, different order of applying the selected structuring elements on the image result in different thinning algorithms. Although the threshold decomposition method provides a way to implement the gray scale erosion and dilation binarily, it is not suitable for implementation in Androx. This is due to the byte addressable nature of the Androx. Thus, a method other than using the threshold decomposition property is presented.

Implementation of the binary morphological operations using the local maximum and minimum method is inefficient because it takes more operations than the parallel SHIFT and AND (or OR) method does. With the library

functions in Androx, the SHIFT operation is in fact implied in the OR (or AND) operation. This fact makes the implementation of the binary morphological operations even more efficient. Although the hit/miss transform is deduced from the erosion, we do not have to start from performing erosion to implement it in the Androx. By implementing the thinning transform using the window operations presented in Chapter 4, the thinning algorithms can be easily implemented without performing a sequence of the operations on the image and the complement of the image. Using the Androx library functions, the gray scale erosion (dilation) can be implemented by simply adding (subtracting) the values of two windows and then writing the maximum (minimum) to the corresponding position. This method makes the gray scale operation not only simpler but also more efficient than the threshold decomposition method.

Reference

[1] *Androx ICS Library Programmer's Reference Manual,* Revision 4.0, Androx Corporation.

[2] C. R. Giardina and E. R. Dougherty, *Morphological Methods in Image and Signal Processing,* Prentice-Hall, Inc., 1988.

[3] J. Serra, *Image Analysis and Mathematical Morphology,* New York: Academic, 1982.

[4] Y. Chang and Nirwan Ansari, " A Practical VLSI Realization of Morphological Operations," Submitted to *IEEE Transactions on Circuits and Systems for Video Technology.*

[5] Y. Nakagawa and A. Rosenfeld, "A Note on The Use of Local Min and Max Operations in Digital Picture Processing," *IEEE Trans. Syst., Man, Cybern.,* vol. SMC-8, pp.632-635, Aug. 1978.

[6] S.R. Sternberg, "Parallel Architectures for Image Processing," in *Proc. 3rd IEEE Int. Comput. Software Appl. Conf.,* Chicago, IL, 1979, pp. 712-717.

[7] B. Jang and R. L. Chin, "Analysis of Thinning Algorithms Using Mathematical Morphology," *IEEE Trans. Pattern Anal. Machine Intell.,* Vol. 12, No. 6, June 1990, pp. 541-551.

[8] P. A. Maragos, "Morphological Skeleton Representation and Coding of Binary images, " *IEEE Trans. Acoust., Speech, Signal Processing,* Vol. Assp-34, No. 5, October 1986, pp. 1228-1244.