AN ABSTRACT OF THE THESIS OF

Jin-Woo Eo for the degree of Master of Science in Electrical and Computer Engineering presented on June 9, 1988.

Title: Mathematical Morphology: A Case Study of Linear Shift-Invariant Filter Implementation

Abstract approved: Redacted for Privacy
W. J. Kolodziej

Morphological filters have shown good performance in various image processing applications. Furthermore, their algorithms are easy to implement in special architectures, with the exception that linear filtering cannot be implemented in many of these cases.

In this study, a new architecture and corresponding algorithm, capable of realizing linear shift-invariant filters, as well as morphological filters, are presented, using the symmetrical nature of the most commonly used linear filtering masks. The architecture suggested in the study, along with the algorithm presented, will provide considerable improvement in the power of image processing systems, as well as improvements in the application fields of mathematical morphology.

Mathematical Morphology: A Case Study of Linear Shift-Invariant Filter Implementation

by

Jin-Woo Eo

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed June 9, 1988

Commencement June 1989

APPROVED:

Redacted for Privacy

06-20-1988

Associate Professor of Electrical and Computer Engineering in charge of major

Redacted for Privacy

Head of Department of Electrical and Computer Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented June 9, 1988

Typed by B. McMechan for Jin-Woo Eo

Table of Contents

Table of Contents (continued)

List of Figures

List of Figures (continued)

MATHEMATICAL MORPHOLOGY: A CASE STUDY OF LINEAR SHIFT-INVARIANT FILTER IMPLEMENTATION

I. INTRODUCTION

1.1 Historical Overview of Mathematical Morphology

Mathematical morphology was developed in 1964 at the Paris School of Mines (France) by G. Matheron and J. Serra, who were asked to (1) investigate the relationships between the geometry of porus media and their permeabilities and (2) to quantify the petrography of iron ores in order to predict their milling properties [2]. By probing and transforming a geometric structure with different patterns of predefined shapes, or structuring elements, Matheron and Serra extracted a number of different techniques. Thus, they called their geometric transformations "morphology," meaning the "study of forms." In order to avoid tying the results of morphological transformations to the specific objects of the observer, they derived four mathematical quantification constraints with four corresponding principles, or the method of "mathematical morphology."

Originally, Matheron and Serra represented image objects and structuring elements by sets in a Euclidean

space. Hence, the morphological operations are actually set operations based on unions and intersections. The simplest morphological operations are erosion, dilation, opening, and closing, all of which grew out of Minkowski addition and subtraction. Since binary images can be represented by two-dimensional sets, Matheron [1] and Serra [2] applied the Minkowski set operations to image analysis by mathematical morphology. In this sense, image analysis by mathematical morphology is also called morphological image analysis or morphological filtering.

Binary signal analysis was extended to multilevel signals by Serra [2] and Sternberg [3,4]. Serra used signal cross sections to generalize the morphological operations of multilevel signals. Sternberg further generalized morphological transformations for multilevel signals by considering greytone images as surfaces of three-dimensional volumes (the umbra).

Since Matheron [1] provided a theorem stating that any translation-invariant increasing system can be represented by a union of morphological set erosions, mathematical morphology extended its field to algebraic (not topological) analysis. Lantuejoul and Serra [8] studied properties of generalized (algebraic) openings and closings, which they called M-filters, examining the relationships between morphological filters and classical linear filters. By showing that the mapping

between a function and its umbra is one-to-one, Maragos [10] viewed any system as a set mapping from one class of sets to another, and unifying the representation, analysis, and synthesis of a large class of translation-invariant systems through the concept of minimal elements.

There are numerous applications for morphological filters in image processing and analysis, including biomedical image processing, automated industrial inspection, shape recognition, nonlinear filtering, edge detection, noise suppression, thinning, enhancement, representation and coding, texture analysis, and shape smoothing. The literature for these applications has been surveyed by Maragos [11].

Between 1965 and 1975, Serra [2] constructed four prototype devices, or "texture analyzers," for the performance of morphological operations. However, since the onset of modern computer technology, Serra's texture analyzers can no longer be said to represent the cutting edge of advance techniques and various parallel architectures have been developed. The pipeline image processor (cytocomputer) [23], developed at the Environmental Research Institute of Michigan, has been evaluated as most effective architecture for the performance of mathematical morphology.

1.2 Research Motivation and Organization of the Study

Morphological filters have shown good performance in various image processing applications. Furthermore, their algorithms are simple to implement on special architectures, with the exception that linear filtering (convolution) cannot be implemented in many such cases. However, linear filtering cannot be avoided in many applications. Therefore, it is of interest to develop a practical linear filtering algorithm which can be implemented on morphological transformation-oriented architectures.

Since mathematical morphology has not become a popular genre in either image processing or system theory, a historical overview of theoretical developments, applications, and devices is provided in this chapter. In Chapter II, the philosophy and principles of mathematical morphology and basic morphological transformations are summarized, while Chapter III provides an analysis of morphological filters, in accorance with signals and systems classification, their applications, and the relationship between morphological and linear filters. Chapter IV includes discussion of morphological transformations as neighborhood operations for implementation on parallel architecture and a review of selected effective architecture based upon reasonable

effectiveness criteria, as well as providing a morphological transformation algorithm and a linear filtering algorithm. Conclusions drawn from this study and suggestions for future research are presented in Chapter V.

II. WHAT IS MATHEMATICAL MORPHOLOGY?

The literal meaning of mathematical morphology is the quantitative description of geometrical structures. Since Maragos [10] has provided a thorough analysis of the philosophy and principles of mathematical morphology, originally proposed by Matheron and Serra, the following introduction represents a summary of Maragos' descriptions.

To describe a geometrical structure an image object and an observer are required. According to Matheron [1], by itself the image object contains no information until the observer determines, a priori, which property of the object is to be examined, i.e., an observer sees in an image object only what he wants to see. The problem is how the observer can reach initial assumptions about some aspects of geometrical shapes of which he cannot yet be aware? Serra summarized the problems and the means to a solution as follows:

> The notion of a geometrical structure or texture is not purely objective. It does not exist in the phenomenon itself, nor in the observer, but somewhere in between the two. Mathematical morphology quantifies this intuition by introducing the concept of structuring elements. Chosen by the morphologist, they interact with the object under

> study, modifying its shape and reducing it to a sort of caricature which is more expressive than the actual phenomenon. [2, p. v]

Since there are a large variety of possible object shapes and sizes, number of potential structuring elements which can be offered is extensive. Then the next step is to provide a means of choosing and classifying the elements, or sorting them structurally. Once the image object can be transformed with the use of different structuring element, information about its size, shape, smoothness, and orientation can be obtained. Thus, "only the interaction of the image with the structuring element has an objective meaning" [2, p. 57].

In order that the morphological transformation of an image object can be said to be quantitative, the results of the morphological transformation must not be dependent on the specific viewpoint of the observer. In the system developed by Matheron and Serra, this objectivity is dependent upon four quantification constraints, including:.

C1) The observer cannot know the exact location of the object;

C2) All observed objects can be magnified or reduced;

C3) Due to the field of view of optical instrument or the restrictions of the size which the ob-

serving instrument can provide, only a piecemeal knowledge of object can be extracted; and

C4) Since the various powers of resolution of observing instruments divide the scale of the dimensions, there will always exist such details that cannot be delineated properly.

Corresponding to the these constraints, mathematical morphology presents four supporting principles. To express these four principles, let the set X denote the original image object and $\Phi(X)$ the transformed object. Then, every morphological transformation must satisfy the following four principles:

P1) Translation invariance. The result of the translation should not depend on the location of the object in the space, which may be symbolically written:

$$\Phi(X_h) = [\Phi(X)]_h$$

where X_h denotes the translate of set X by vector h.

P2) Compatibility under change of scale. The transformation should be compatible with scale changes. Then, a following family of transformations, Φ_α, can be generated, depending on the positive parameter α,

$$\Phi_\alpha(X) = \alpha\Phi(X/\alpha) \ .$$

P3) Local knowledge. The transformation Φ satisfies the local approach principle if, for any

bounded set Z' in which we want to know $\Phi(X)$, we can find a bounded set Z in which the knowledge of X is sufficient to locally perform (i.e. within Z') the transformation. Symbolically, for all bounded Z' and for all bounded Z:

$$[\Phi(X \cap Z)] \cap Z' = \Phi(X) \cap Z' \ .$$

P4) Upper semi-continuity. For any increasing set transformation, Φ, and a decreasing sequence of closed sets, X_n, tending towards a limit X, the sequence of transformed sets, $\Phi(X_n)$, must tend towards $\Phi(X)$.

2.1 Minkowski's Set Addition and Subtraction

An m-dimensional (m-D) signal can be represented mathematically by a function of m independent variables (m-D function). This function can be any value. If it is assumed that this function can have only two distinct values, it is called the binary function, in which case the signal can be represented as set in an m-D Euclidean space. Since mathematical morphology is based on the set-theoretical method, the ensuing discussion in this section begins with binary signals (viewed as a set). In section 2.3, this set-theoretical method is expanded into multilevel functions.

2.1.1 Minkowski Addition ⊕ Definition

Minkowski set addition, $A \oplus B$, of two sets A and B, consists of all points that can be expressed as an vector addition, a+b, where the vector a and b belong, respectively, to the sets A and B:

$$A \oplus B = \{a+b: a \in A, b \in B\} . \tag{2-1}$$

From Equation (2-1) the following properties are obtained:

Minkowski addition is an associative and commutative operation , (2-2)

$$A \oplus \{o\} = A , \tag{2-3}$$

and

$$A \oplus \emptyset = \emptyset , \tag{2-4}$$

where {o} is origin.

Let $A \oplus \{b\}$ denote the translate of A by the vector b, often writing A_b instead of $A \oplus \{b\}$. Also we denote B^S as the symmetric set of B with respect to the origin: $B^S = \{-b: b \in B\}$. Thus, following expression is obtained:

$$A \oplus B = \{a+b: a \in A, b \in B\} = \bigcup_{b \in B} A_b = \bigcup_{a \in A} B_a$$

$$= \{z: A \cap (B^S)_z \neq \emptyset\} . \tag{2-5}$$

2.1.2 Dilation Definition

The dilation of A by B is the set $\{z: A \cap B_z \neq \emptyset\}$ of the point z such that A hits the translate B_z. Therefore the dilation of A by B is equal to $A \oplus B^S$.

The Minkowski set subtraction of A by B, denoted as $A \ominus B$, is defined indirectly as the operational dual to Minkowski set addition with respect to complementation:

$$A \ominus B = (A^c \oplus B)^c = \bigcap_{b \in B} A_b = \{z: (B^s)_z \subset A\} . \qquad (2\text{-}6)$$

The operation dual to dilation is called erosion. The erosion of A by B is the set $\{z: B_z \subset A\}$ of the point z such that the translate B_z is included in A and, by Equation (2-6), is equal to $A \ominus B^s$.

2.1.3 Properties of Set Dilation and Erosion

See Figure 2.1.

2.1.3.1 Translation-invariance

Dilation or erosion by a single point, $X \oplus \{x\}$ or $X \ominus \{x\}$, is the translate of X by x, X_x. For the origin $\{o\}$, $X \oplus \{o\}$ or $X \ominus \{o\}$ is X itself. Consequently, the origin can be taken by any point which is or is not contained by structuring element B. By the relations (2-5) and (2-6), the erosion $X \ominus B$ or the dilation $X \oplus B$ will be the same modulo a shift, which means translation-invariant.

2.1.3.2 Distributivity

Since we can write

$$X \oplus (A \cup B) = \bigcup_{x \in X} (A_x \cup B_x) = (\bigcup_{x \in X} A_x) \cup (\bigcup_{x \in X} B_x) ,$$

we obtain relation (2-7), and by duality (2-8) and (2-9):

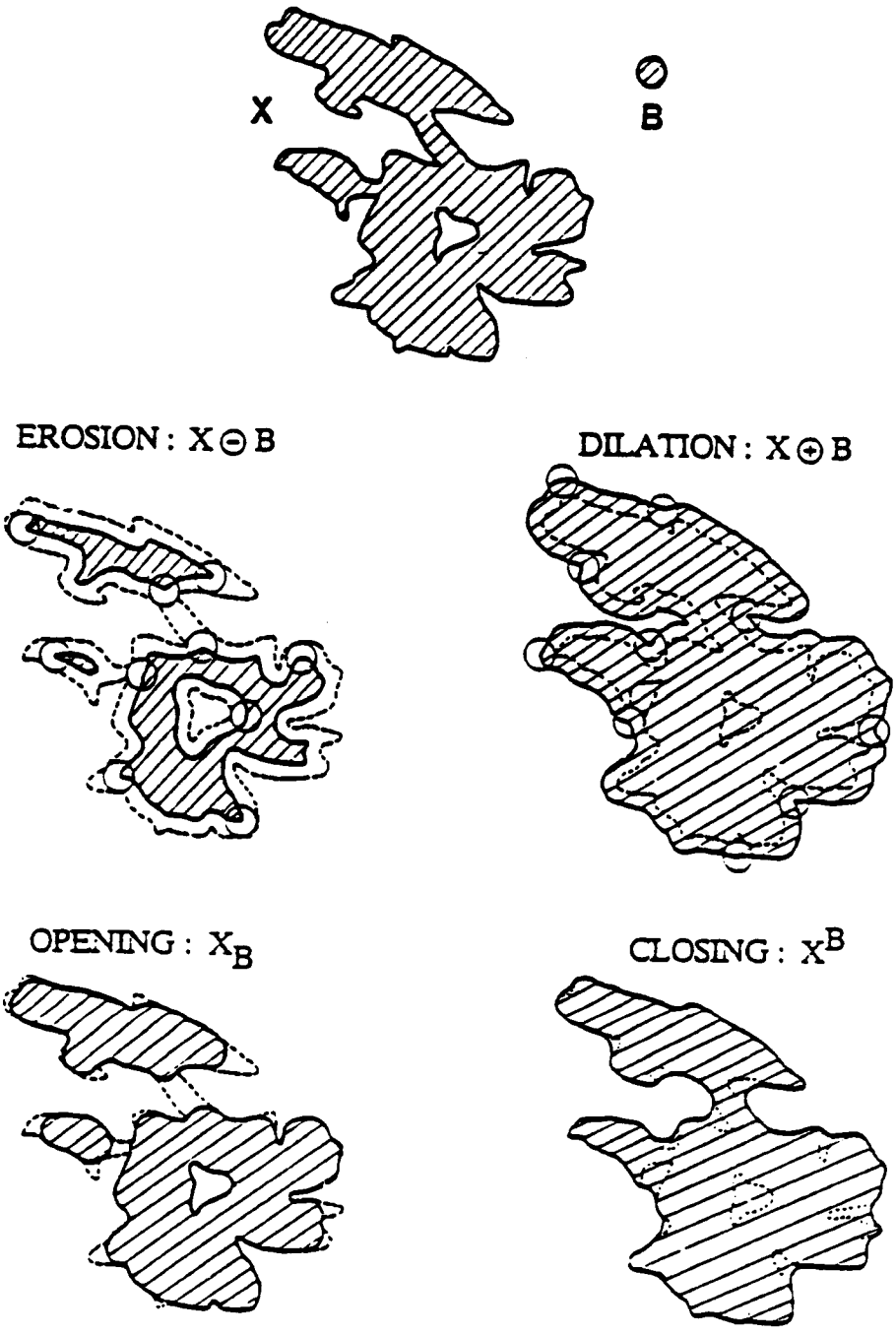


Figure 2.1 Erosion, dilation, opening, and closing of X by B.

$$X \oplus (A \cup B) = (X \oplus A) \cup (X \oplus B) , \qquad (2\text{-}7)$$

$$X \ominus (A \cup B) = (X \ominus A) \cap (X \ominus B) , \qquad (2\text{-}8)$$

and

$$(X \cap Z) \ominus B = (X \ominus B) \cap (Z \ominus B) . \qquad (2\text{-}9)$$

The above three relations imply a measure of technological importance. From (2-7) and (2-8), X can be dilated or eroded by taking the structuring element piece by piece, and then combining the intermediary results, respectively, by union or intersection.

2.1.3.3 Iterativity

From (2-5) and (2-6) we derive:

$$(X \oplus A) \ominus B = \bigcap_{b \in B} (X \oplus A)_b = \bigcap_{b \in B} \bigcup_{a \in A} X_{a+b} , \qquad (2\text{-}10a)$$

$$(X \ominus B) \oplus A = \bigcup_{a \in A} (X \ominus B)_a = \bigcup_{a \in A} \bigcap_{b \in B} X_{a+b} , \qquad (2\text{-}10b)$$

and thus we obtain the set inequality:

$$(X \ominus B) \oplus A \subset (X \oplus A) \ominus B . \qquad (2\text{-}11)$$

This means that it is more severe to erode before dilating than to do the reverse.

Consider the following successive erosions:

$$(X \ominus A) \ominus B = \bigcap_{b \in B} (X \ominus A)_b = \bigcap_{b \in B} \bigcap_{a \in A} X_{a+b} = \bigcap_{z \in A \oplus B} X_z ,$$

then

$$(X \ominus A) \ominus B = X \ominus (A \oplus B) . \qquad (2\text{-}12)$$

Taking complements from (2-12):

$$(X \oplus A) \oplus B = X \oplus (A \oplus B) . \qquad (2\text{-}13)$$

Relations (2-12) and (2-13) provide a powerful morphological tool, which enables decomposition of a given

structuring element into the Minkowski sum of several much simpler segments. Figure 2.2 shows that the hexagon structuring element can be considered as the Minkowski sum of the segments.

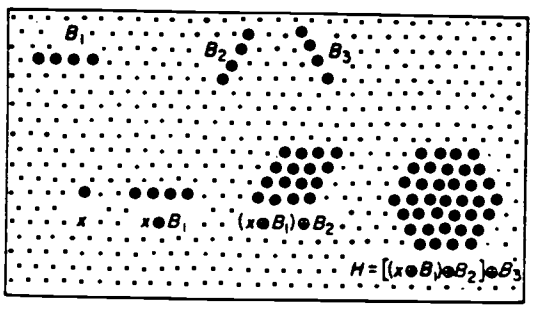


Figure 2.2 The hexagon considered as the Minkowski sum of the segments (adapted from Serra).

2.2 Opening and Closing

By combining erosion and dilation, opening and closing operations are obtained. The opening A_B of A by B and closing A^B of A by B are defined as follows:

$$A_B = (A \ominus B^S) \oplus B$$
$$A^B = (A \oplus B^S) \ominus B \quad . \qquad (2\text{-}14)$$

Since the opening of A by B is the union of the translates, B_y included in A (i.e., $A_B = \cup \{B_y: y \in E, B_y \subset A\}$), this operation smooths the contours of X, suppresses the small islands and the sharp capes of X, and cuts the narrow isthmuses. By duality, since the closing of an object, X, is equivalent to the opening of its background, the closing blocks up the narrow channels, the small lakes, and the long thin gulfs of X (see Figure

2.1). The morphological opening defined by (2-14) has the following three properties:

1. Property 1: Anti-extensivity. Since every point $x \in X_B$ belongs to at least one B included in X, X_B is included in X:

$$X_B \subset X \ . \qquad (2\text{-}15a)$$

2. Property 2: Increasing property. Let X be included in X' ($X \subset X'$). Since every point $x \in X_B$ belongs to one B contained in X, so in X', the open set X_B is contained in X'_B:

$$X \subset X' \rightarrow X_B \subset X'_B \ . \qquad (2\text{-}15b)$$

3. Property 3: Idempotence. By anti-extensivity we can write:

$$(X_B)_B \subset X_B \qquad \text{--------- (*)}$$

and $(X_B)_B$ can be written as follows:

$$(X_B)_B = \{[(X \ominus B^s) \oplus B] \ominus B^s\} \oplus B \ .$$

We can then write $(X \ominus B^s) \oplus B \ominus B^s = (X \ominus B^s)^{B^s}$, and

$$(X \ominus B^s)^{B^s} \supset (X \ominus B^s)$$

by the extensivity of the closing. Since

$$(X_B)_B = (X \ominus B^s)^{B^s} \oplus B \supset (X \ominus B^s) \oplus B = X_B \text{ -------- (**)} \ .$$

By (*) and (**), $(X_B)_B = X_B$. (2-15c)

By duality, the closing X^B is extensive, increasing, and idempotent:

$$X^B \supset X \ , \qquad (2\text{-}16a)$$

$$X' \subset X \rightarrow X'^B \subset X^B \ , \qquad (2\text{-}16b)$$

and

$$(X^B)^B = X^B \quad . \tag{2-16c}$$

2.3 Relationship between Sets and Functions

To extend all the morphological set transformations to functions, relationships must be established between sets and functions. Therefore, representing functions by sets is the main issue. There are two different but equivalent approaches to this problem: (1) An m-D function can be represented either by an ensemble of m-D sets called its cross sections or by (2) a single (m+1)-D set called its umbra. Figure 2.3 shows a 1-D function f, one of its cross-sections, and its umbra.

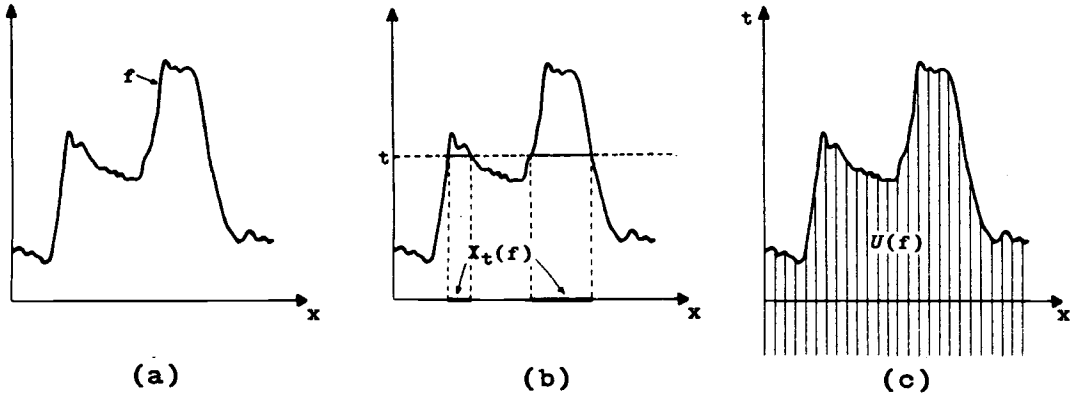


Figure 2.3 (a) A function f, (b) cross-section $X_t(f)$, and (c) umbra $U(f)$.

2.3.1 Cross-Section of a Function

The cross-section $X_t(f)$ of f at level t is defined by the set obtained by thresholding f at level t:

$$X_t(f) = \{x \in D: f(x) \geq t\}, \quad -\infty < t < \infty . \qquad (2\text{-}17)$$

Since work is carried out in the class of closed sets, it is obvious that all cross-sections of f should be closed. There is a useful classical topological result which may be applied:

2.3.1.1 Theorem 2-1 [29]

A real-valued function f on R^m is upper semi-continuous (u.s.c.) if and only if (iff) its cross-sections $X_t(f)$ are closed sets in R^m for all $t \in R$. Consequently, the corresponding class of function which is always treated is the class of u.s.c. functions on D, denoted as USC(D).

If all the cross sections of a u.s.c. function are known, it can be uniquely reconstructed by Theorem 2.

2.3.1.2 Theorem 2-2 [2]

First, let f(x) be a u.s.c. real valued function on R^m and let $X_t(f)$, $t \in R$ be its cross sections. Then, the X_t's are closed and monotonically decreasing, i.e.

$$t_1 < t_2 \rightarrow X_{t1} \supset X_{t2} \text{ and } X_t = \bigcap_{r<t} X_r \qquad (2\text{-}18)$$

and we have

$$f(x)=\sup\{t \in R: x \in X_t\} . \qquad (2\text{-}19)$$

Conversely, a family X_t $(-\infty \leq t \leq +\infty)$ of closed sets generates a u.s.c. function, f(x), if and only if conditions (2-18) are satisfied; then f(x) is defined by (2-19) and satisfies (2-17).

The main concept underlying Theorem 2-2 is illustrated in Figure 2.4, showing a 1-D function f(x) and two of its cross sections at level t_1 and t_2. For a given point $x \in R$, $x \in X_{t1}(f)$ iff $f(x) \geq t_1$. In contrast, $x \in X_{t2}(f)$ iff $f(x) < t_2$. Thus, the value of f at x is equal to the supremum of t's such that $f(x) \geq t$ or equivalently, $x \in X_t(f)$.

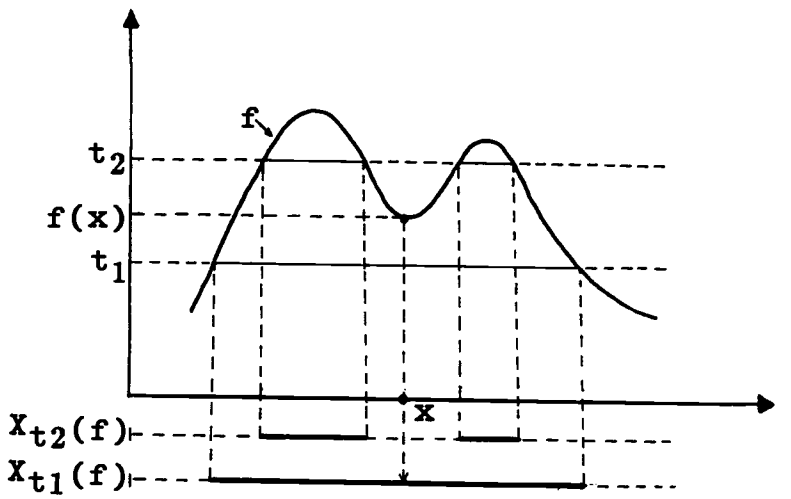


Figure 2.4 Reconstruction of a function from its cross section.

2.3.2 Umbra of a Function

The concept of cross section deals with sets which have same dimensions, (m-D), as function. Now, the morphology of sets in Euclidean (m+1)-D space is considered. In illustration, it can be assumed that the function is a 2-D image without loss of generality. The cross sections of this image can be considered as binary images of Euclidean 2-D, seen as flat cutouts of the horizontal X,Y plane, henceforth referred to as the

binary plane. On the other hand the umbra, which is considered in Euclidean 3-D, has a solid volume. The details of an umbra's third dimension can be determined by a single parameter, the height of t at coordinates (x,y) of the binary plane.

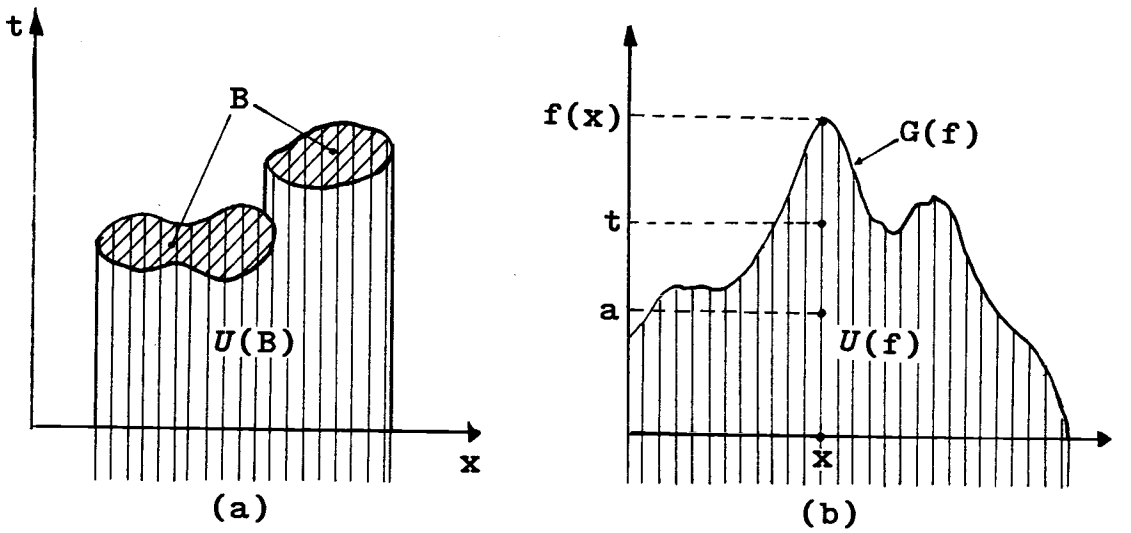


Figure 2.5. (a) Umbra of a set B, (b) umbra and graph of a function.

Umbra means shadow, and the umbra of a set, B, in 3-D includes both B and the volume of the point in its shadow. The shadow is cast by a point light source at an infinite distance in the positive T direction. Figure 2.5a shows this concept. Analytically, the umbra of the set B can be expressed as a morphological transformation of B and a structuring element, T, composed of the points belonging to the T axis, including the origin. By Minkowski-sum definition (section 2.1), the umbra of the closed set B equals $B \oplus (-\infty, 0]$. This relation corresponds to $B \oplus [0,\infty)^S$, dilating B by the posi-

tive T axis. Similarly, the umbra of a function, f, is the Minkowski-sum of the function and $(-\infty,0]$. Thus the umbra of f, denoted as $U(f)$, is the set:

$$U(f) = \{(x,t): f(x) \geq t\} . \qquad (2\text{-}20)$$

According to Serra [2], mapping a u.s.c. function f and $U(f)$ is one-to-one. However, Maragos [10] mathematically formalized what Serra had maintained with a proof.

2.3.2.1 Theorem 2-3 [10]

First, to any real-valued u.s.c. function, f(x), $x \in R^m$, there corresponds a unique umbra $U(f)$. This umbra, U, is a closed set in R^{m+1} such that

$$(x,t) \in U \leftrightarrow t \leq f(x) \leftrightarrow x \in X_t(f) \qquad (2\text{-}21)$$

and

$$(x,t) \in U \rightarrow (x,a) \in U, \text{ for all } a < t . \qquad (2\text{-}22)$$

Also for each $x \in R^m$,

$$f(x) = \sup\{t \in R: (x,t) \in U\} \qquad (2\text{-}23)$$

b) Conversely, to any closed subset U of R^{m+1} satisfying (2-22), there corresponds a unique u.s.c. function f(x), which can be constructed from (2-23). Thus the umbra of f(x) is equal to U.

III. MORPHOLOGICAL FILTERS

Morphological transformations locally modify geometric features of signals by introducing the concept of structuring elements. In the sense of modifying signals, this transformation can be termed a filter. In section 2.1, function and set were introduced to represent signals mathematically, with the distinction that an m-D function implies a multilevel m-D signal, whereas an m-D set refers to a binary m-D signal. Maragos [10] introduced a classification of systems in accordance with the classification of signals (e.g., function or set). Since the term "system" means any process or device responding to particular signals by producing other signals, filters correspond to systems.

According to Maragos' [10] classification, an m-D set-processing (SP) filter is a filter capable of accepting m-D binary signals as inputs and producing m-D binary signals as outputs. An m-D function processing (FP) filter is any filter capable of accepting m-D functions as inputs and producing m-D functions as outputs. A sub-class of m-D FP filters can produce an m-D binary signal whenever the input is also an m-D binary signal; these are called function-and-set processing

(FSP) filters. SP filters were discussed in sections 2.1 and 2.2. In this chapter, FSP and FP filters are discussed, using the relationship between sets and functions (refer to section 2.3). In addition, some of the applications of morphological filters are surveyed, and the relationship between morphological and linear filters, according to Maragos [10], are discussed.

3.1 Morphological Filters

3.1.1 Morphological Transformation of Function by Set

Nakagawa and Rosenfeld [6] have shown that if local minimum and maximum operations are applied to a picture (function) Π, and Π is then thresholded (using any threshold t), the results are exactly same as if Π were first thresholded at t and shrinking and expanding operations were then applied to the thresholded Π (i.e., local min and max commute with thresholding). The terms shrinking and expanding are used in the field of fuzzy logic, which was introduced by Zadeh in 1965, and they have,respectively, meanings identical to those of SP erosion and dilation. Using this property, Goetcherian [7] extended some already well known binary processes into grey level algorithms.

Consider a morphological transformation, Φ, of function, f, by a structuring element B. This trans-

formation, Φ, is basically an FP system. Since the FP system Φ commutes with thresholding, the following relationship is obtained:

$$X_t[\Phi(f)] = \Phi[X_t(f)] \text{ for all } t \in R \ . \qquad (3\text{-}1)$$

The right side of relation (3-1) shows that Φ can also be an SP system for all cross-sections, $X_t(f)$. Therefore, the morphological transformations of functions by sets are FSP systems. The simplest of these filters are the erosion, $f \ominus B^s$, the dilation, $f \oplus B^s$, the opening, f_B, and the closing, f^B, of an u.s.c. function f by a compact structuring set B. The following relations (Table 3.1) show the analytical definitions operating on function f as a whole and on each of its cross-sections:

Table 3.1 Analytical definitions of function f.

	Functions	Cross-sections (for all t)	Eqn.
Erosion:	$(f \ominus B^s)(x) = \inf\{f(y): y \in B_x\}$	$X_t(f \ominus B^s) = X_t(f) \ominus B^s$	(3-2a)
Dilation:	$(f \oplus B^s)(x) = \sup\{f(y): y \in B_x\}$	$X_t(f \oplus B^s) = X_t(f) \oplus B^s$	(3-2b)
Opening:	$(f_B)(x) = [(f \ominus B^s) \oplus B](x)$	$X_t(f_B) = [X_t(f)]_B$	(3-2c)
Closing:	$(f^B)(x) = [(f \oplus B^s) \ominus B](x)$	$X_t(f^B) = [X_t(f)]^B$	(3-2d)

The FSP erosion or the FSP dilation of f by B at any point x is obtained by taking, respectively, the

infimum or the supremum of f inside the shifted set B_x, which plays the same role as a running window in signal processing. Maragos [10] provided proof for Equation (3-2), using the relation (2-19), allowing reconstruction of the whole transformed function. For a discrete system, infimum or supremum operations should be replaced,respectively, by local minimum or maximum.

Figure 3.1 shows erosion and dilation of the 1-D sampled function f by the symmetric set B{-2,-1,0,1,2}. It may be seen that the erosion of a function by a set reduces the peaks and enlarges the minima of the function. The dilation of f by B increases the valleys and enlarges the maxima of the function. Figure 3.2 shows the opening and closing of the function by same set B. As expected, the opening smooths the graph of function f from below by cutting down its peaks and the closing smooths the graph of function f from above by filling up its valleys. Goetcherian [7] noted that iterative erosion and dilation can provide non-linear low-pass filtering. Figure 3.2c shows the low-pass filtering effect after double erosion and double dilation ($f\ominus B\ominus B\oplus B\oplus B$), i.e., the higher the number of iterations, the lower the cut-off frequencies.

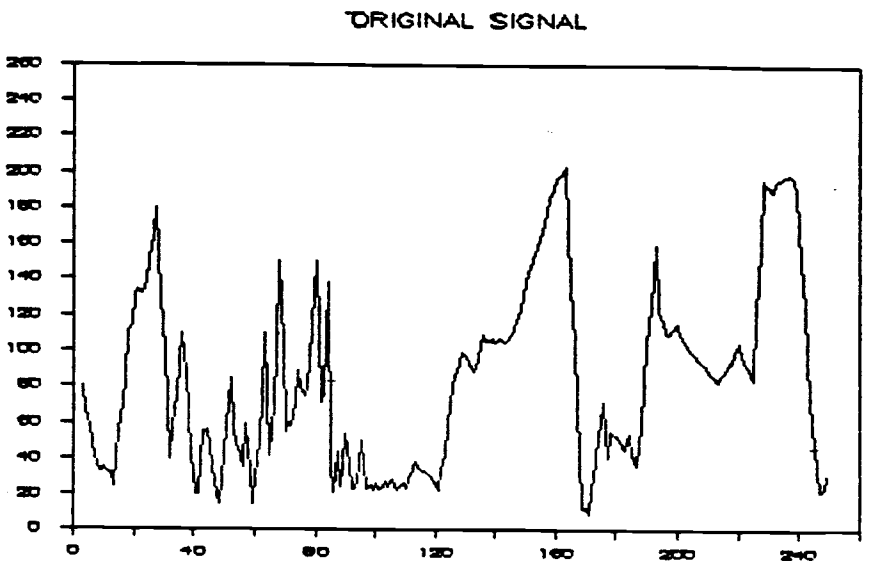


Figure 3.1a Original signal f.

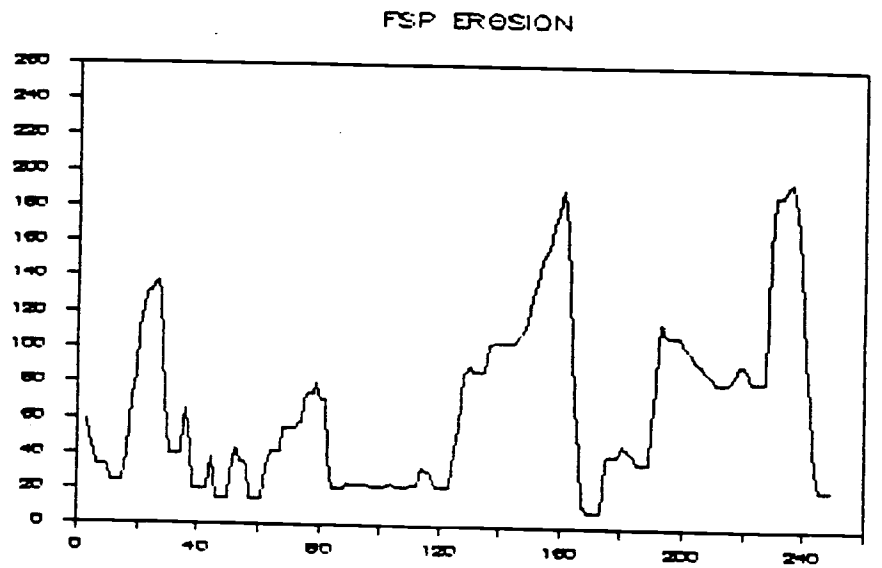


Figure 3.1b FSP erosion of f by a set B (B = {-2,-1,0,1,2}).

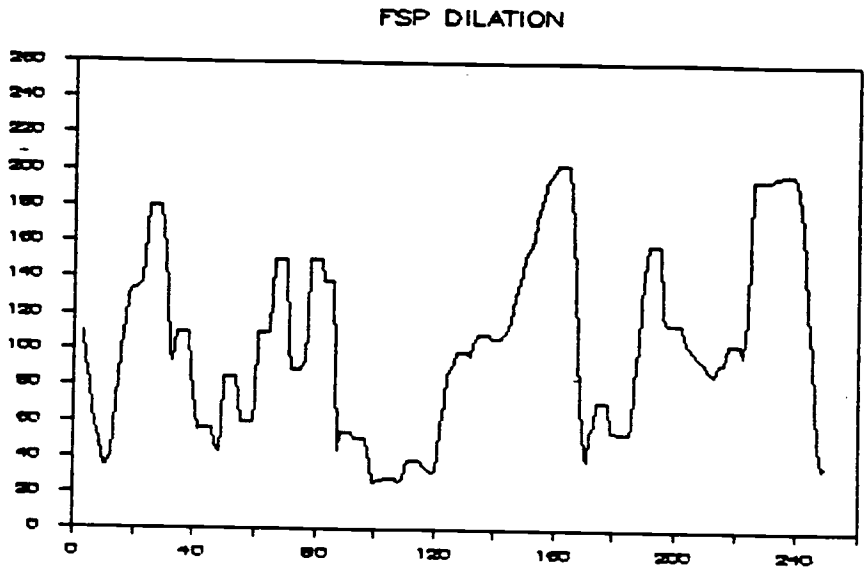


Figure 3.1c FSP dilation of f by a set B.

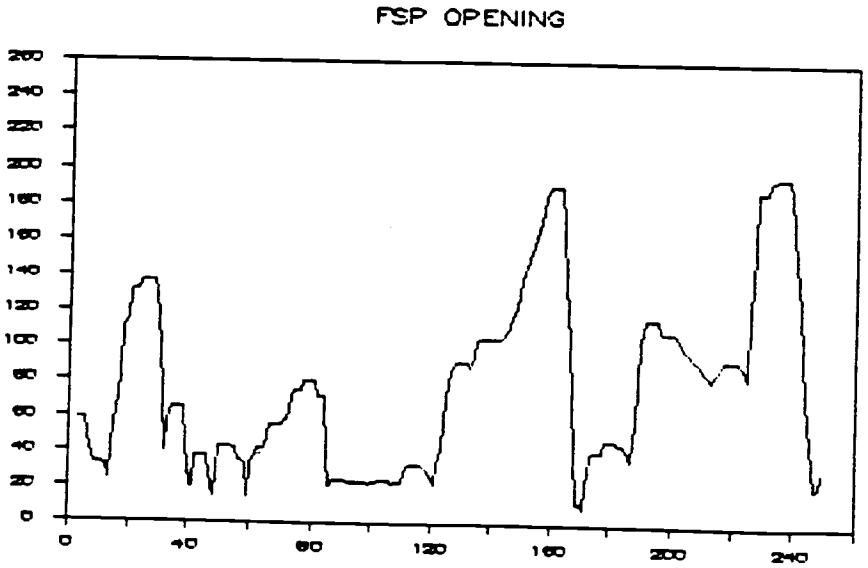


Figure 3.2a FSP opening $X_B = X \ominus B^S \oplus B$.

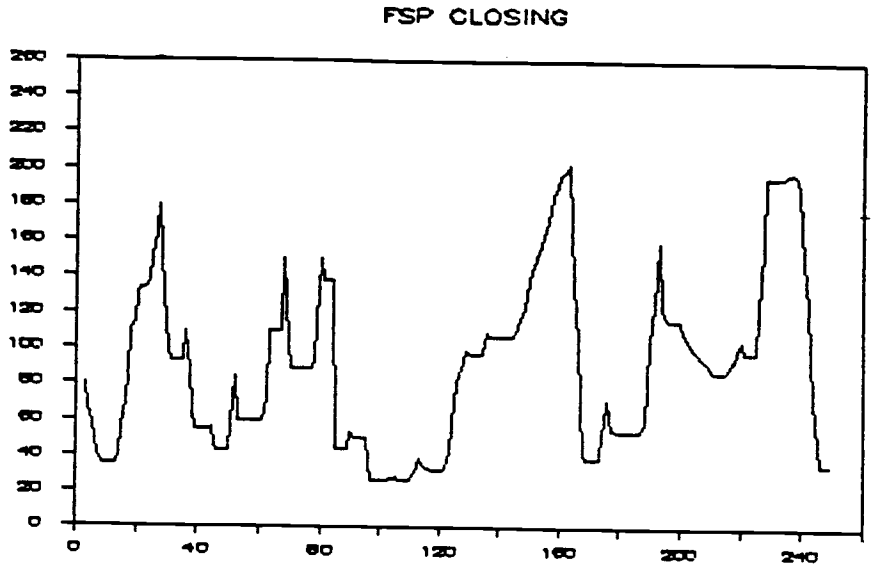


Figure 3.2b FSP closing $X^B = X \oplus B^S \ominus B$.

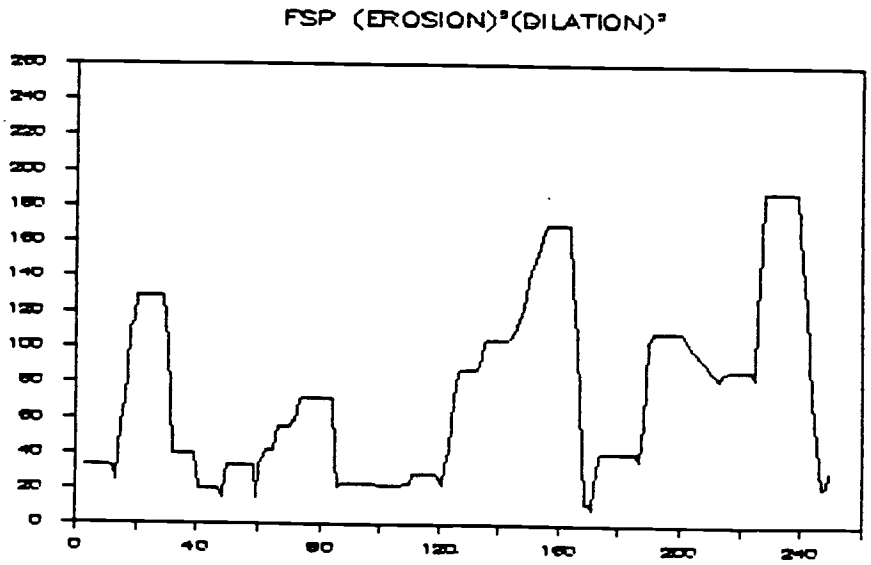


Figure 3.2c FSP low pass filter, $X \ominus 2B \oplus 2B$.

3.1.2 Morphological Transformation of Function by Function

An u.s.c. function, f, can be transformed by another u.s.c. function, g, the structuring function, which has a compact region of support. In this transformation, the cross-section approach is more complicated than the umbra approach. Therefore, discussion is limited to only the umbra interpretation.

Serra [2] gives the following analytic relations for the Minkowski FSP addition f⊕B and subtraction f⊖B of the u.s.c. function, f, with a compact structuring element B:

$$U(f \oplus B) = U(f) \oplus B = U(f) \oplus U(B) \qquad (3\text{-}3a)$$

and

$$U(f \ominus B) = U(f) \ominus B^r = U(f) \ominus [U(B)]^r \qquad (3\text{-}3b)$$

where $B^r = \{(x,-t) \in D \times R: (x,t) \in B\}$ is the reflected set of B with respect to the horizontal plane, D, of abscissas x. This FSP transformation is a special case of function (f) transformation by a function g. Since $U(g)$ may be considered a set, we replace B with $U(g)$ in Equations (3-3a) and (3-3b). Then the umbra $U(f \oplus g)$ of the Minkowski function addition f⊕g of f and g is equal to the Minkowski set addition of $U(f) \oplus U(g)$. Similarly, the umbra $U(f \ominus g)$ of the Minkowski function subtraction f⊖g of g from f is equal to the Minkowski set subtraction of $[U(g)]^r$ from $U(f)$. As mentioned in section

2.2, the dilation or the erosion of function f by structuring function g is, respectively, equal to the Minkowski sum, $f \oplus g^S$, or subtraction, $f \ominus g^S$.

From this point, a graphical illustration (Figure 3.3) is presented, which may be used to extract algebraic formulations. Consider the umbras $U(f)$ and $U(g)$ of 1-D functions f and g.

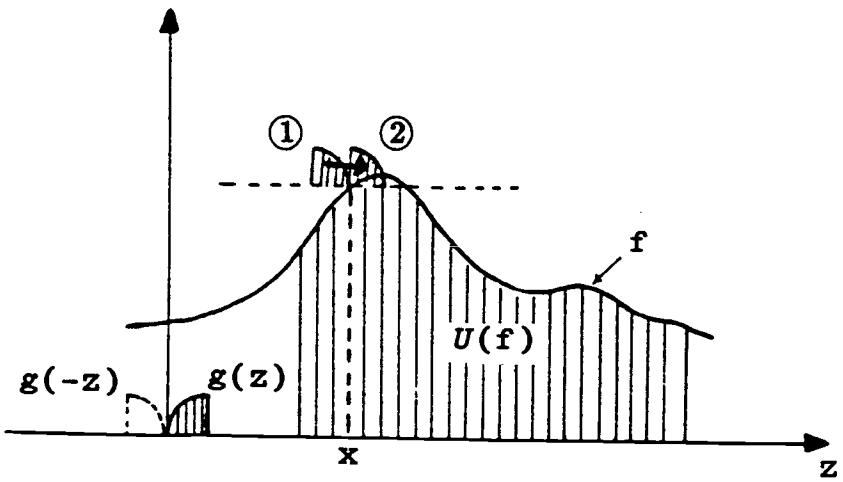


Figure 3.3 FP transformation procedure.

Then the umbras $U(f)$ and $U(g)$ are 2-D sets and the dilation $U(f) \oplus U(g^S)$ is the set dilation. Therefore, this set transformation can be illustrated by defining the set dilation in accordance with the following procedure:

1) Take the symmetric structuring function ($g^S = g(-z)$) and its umbra; (the upper part of abscissa z of the original umbra can be taken as umbra (Fig. 3.3) without loss of generality);

2) Shift the umbra $U(g^S)$ parallel in order that the right end point of $U(g^S)$ can hit the f(x) point (at Fig. 3.3, ① position);

3) Calculate the sum of f and g^S over the region of support of g (e.g., f(x-z) + g(-z));

4) While shifting the umbra $U(g^S)$ to the right side until the left end point of $U(g^S)$ hits the f(x) point (at Fig. 3.3, ② position), then repeat procedure step 3; and

5) The supremum of the results of procedure step 3 is the value of the dilation $(f \oplus g^S)(x)$.

This procedure can be formulated by the following algebraic equation,

$$(f \oplus g^S)(x) = f(x) \oplus g(-x) = \sup_{z \in D} \{f(x-z) + g(-z)\}$$

or equivalently,

$$(f \oplus g^S)(x) = \sup_{z \in D} \{f(z) + g(z-x)\} \; . \qquad (3\text{-}4a)$$

Similarly, the erosion of a function f by structuring function g can be represented:

$$(f \ominus g^S)(x) = \inf_{z \in D} \{f(z) - g(z-x)\} \; . \qquad (3\text{-}4b)$$

As may be seen in Equations (3.4a) and (3.4b), the dilation, or the erosion, of f by g is defined, respectively, through an additive, or a subtractive, convolution between f and g, then taking a supremum, or an infimum. Maragos [10] referred to this dilation and erosion of two functions as morphological convolutions.

The set, D, is a discrete or continuous space in which f and g are defined. By letting the value of f and g be equal to $-\infty$ wherever they are not defined, it may be assumed that both f and g are defined throughout D. The region of support of the function f is defined, and denoted as Ros(f), as the subset of D in which $f(x) \neq -\infty$. In morphological convolution (3-4), the symbol $(-\infty)$ plays the role of a neutral element in linear convolution.

3.2 Applications of Morphological Filters

A variety of tasks in image processing/analysis can be approached or solved by the use of morphological concepts, and many algorithms can be expressed compactly and systematically in terms of morphological filters. Recently, mathematical morphology has been applied to standard areas of image processing and analysis, including non-linear image filtering, edge detection, noise suppression, shape representation, smoothing, and recognition, skeletonization, and coding. A detailed discussion of each of these applications is beyond the scope of this study, but a brief review of those applications relevant to this study is provided in this section. For a detailed discussion of skeletonization and shape representation, refer to Maragos [14].

3.2.1 Non-linear Filtering

As noted in section 3.1.1, Goetcherian [7] provided the following FS processing morphological filtering operations, using the opening concept:

Let B be the structuring element (set) and

$nB = B \oplus B \oplus \ldots \oplus B$ (n times)

denoting an element of size n, where n is any nonnegative integer. Then,

"Low Pass" = f_{nB} (= $f \ominus nB \oplus nB$)

"High Pass" = $f - f_{nB}$

"Band Pass" = $f_{nB} - f_{mB}$ ($n<m$)

The characteristics of all non-linear filters given above are determined by those of the low pass (e.g., the high pass and the band pass are effectively defined by the low pass filter), which is to say, the higher the n, the lower the cut-off frequency. On the other hand, by combining the structuring elements, there may be other alternative configurations, e.g.:

"Low Pass" = $f \ominus nB \oplus 2nB \ominus nB$, or

"Low Pass" = $f \oplus nB \ominus 2nB \oplus nB$.

3.2.2 Edge Detection

Consider a 2-D image set, X, an image function f, and a structuring element, B. Since SP erosion or FSP erosion erodes the original image, $X-(X \ominus nB)$ gives the boundary of a binary image, X, and $f-(f \ominus nB)$ enhances

the edges of a greytone image, f. The size n of nB controls the thickness of the edge markers.

3.2.3 Noise Suppression

The opening of a function cuts down its peaks, while the closing of a function fills in the valleys. In greytone image, positive noise spikes (bright spots) are peaks, and geographical valleys in the umbra of the function can be considered as negative noise spikes (dark spots). In images these positive and negative noise spikes are called salt-and-pepper noise. Therefore, opening followed by closing (by the same structuring element), which is called open-closing, or closing followed by opening, which is called close-opening, can eliminate salt-and-pepper noise. Median filtering [16] (which is not morphological filtering) can also eliminate this kind of noise. Maragos [10] compared open-closing with median filtering to the same salt-and-pepper noised image. The structuring element for open-closing was 2 x 2 square and the running window for median filtering was 3 x 3 square. Since Maragos obtained a similar result, it was claimed that open-closing has less computational complexity than median filtering for the elimination of salt-and-pepper noise and open-closing can be advantageously used to decompose noise suppression into two separate tasks: cleaning of the positive or cleaning of the negative spikes.

3.3 Morphological Representation for Translation Invariant Systems

Matheron [1] and Maragos [10], respectively, provided morphological representations for any increasing translation invariant set-processing system and any function-processing system. Therefore, the terms translation invariant (TI) and increasing are of particular importance in this section.

As noted in chapter II, TI means that the translation should not depend on the location of object in the space. That is,

$$\Phi(X_h) = [\Phi(X)]_h$$

where X_h denotes the translate of set X by vector h, and Φ denotes the system. This definition can be extended to the function-processing system by considering the umbra of a function as the input X. If the umbra $U(f)$ and $U[\Phi_f(f)]$ of the input and output functions are considered, then the umbra-processing system Φ can be defined as

$$\Phi[U(f)] = U[\Phi_f(f)] \qquad (3\text{-}5)$$

where subscript f denotes the function-processing system. Since the umbra-processing system is a set-processing system, the umbra-processing system Φ is TI if and only if $\Phi(U_z) = [\Phi(U)]_z$. There is a one-to-one and onto mapping between the function-processing system

Φ_f and its equivalent umbra-processing system Φ through the Equation (3-5). Therefore, the function-processing system Φ_f can be defined TI if and only if its equivalent umbra-processing system Φ is TI. That is, the vector translation of the umbra $U(f)$ of f by the vector, $z = (y,t) \in D \times R$ induces the following vector translation of function f(x) by the vector z:

$$f(x) \rightarrow f(x-y) + t,$$

where D is the space where the function f is defined and R is the additional one dimension according to taking of an umbra. Thus, the function-processing system, Φ_f, is called TI if and only if it commutes with such a translation of its input function, i.e.,

$$\Phi_f \text{ is TI} \iff \Phi_f[f(x-y) + t] = [\Phi_f(f)](x-y) + t \ . \qquad (3\text{-}6)$$

The relation (3-6) shows that the vector translation of a function is a shift, both with respect to the argument of a function as well as with respect to its amplitude. On the other hand, the term shift-invariant for a system is only concerned with a shift of the argument of the input function. Hence, a TI function-processing system is also shift-invariant, but the converse is not true.

A system is termed increasing, provided that it preserves the order between inputs and outputs with respect to a certain ordering relationship, i.e., $A \subseteq B$

implies $\Phi(A) \subseteq \Phi(B)$, and similarly for functions, $f \leq g$ implies $\Phi_f(f) \leq \Phi_f(g)$.

3.3.1 Kernel Representation of TI system

Let $E = R^m$ or Z^m be a Euclidean space and let ϑ denote a subcollection of $\rho(E)$ closed under translation, where $\rho(E)$ is the space of all subsets of E. Matheron [1] defined the kernel of translation invariant set-processing system Φ as follows:

$$\kappa = \{A \in \vartheta : o \in \Phi(A)\} , \tag{3-7}$$

where "o" denotes the origin in the space E. That is, the kernel of Φ is a collection of input sets, such that their respective output sets from the system contain the origin. Conversely, if an arbitrary kernel family ,κ, is known, the system Φ is uniquely defined by

$$\Phi(A) = \{x \in E : A_{-x} \in \kappa\} , \quad A \in \vartheta . \tag{3-8}$$

For example, the erosion $\Phi(A) = A \ominus B^s$ by a fixed $B \subseteq E$ has the kernel

$$\xi(B) = \{A \in \vartheta : A \supset B\} . \tag{3-9}$$

We can prove Equation (3-9) using the definition of erosion (2-6). Also, increasing system means that $B \in \kappa$ and $A \supset B$ implies $A \in \kappa$. With the notion of Equation (3-9),

$$\Phi \text{ increasing} \iff \kappa = \bigcup_{B \in \kappa} \xi(B) . \tag{3-10}$$

If Equations (3-9) and (3-10) are compared, it may be concluded that any increasing TI system admits the representation

$$\Phi(A) = \bigcup_{B\in\kappa} (A\ominus B^S) , \qquad (3\text{-}11)$$

that is, $\Phi(A)$ is the union of erosion of A by the sets $B\in\kappa$.

Maragos [10] extended this set-processing representation, Equation (3-11), to the function-processing system using the umbra concept. If we consider a function, f, and a function-processing system, Φ_f, then its umbra, $U(f)$, belongs to the kernel of Φ if and only if $0\in\Phi[U(f)]$. From Equation (3-5),

$$U(f)\in\kappa(\Phi) \iff (0,0) \in U[\Phi_f(f)] . \qquad (3\text{-}12)$$

From the umbra property, Equation (3-12) means $[\Phi_f(f)](0) \geq 0$. Thus, the kernel of any TI function-processing system, Φ_f, can be defined as

$$\kappa(\Phi_f) = \{f\in\vartheta: [\Phi_f(f)](0) \geq 0\} . \qquad (3\text{-}13)$$

That is, the kernel of Φ_f is a collection of input function, f, whose output function $\Phi_f(f)$ at the origin has a nonnegative value. The erosion $\Phi_f(f) = f\ominus g^S$ by a function, g, is $\inf\{f(z)-g(z-x):z\in D\}$ (Equation (3-4b)). Thus, $f\in\kappa(\Phi_f)$ implies

$$(f\ominus g^S)(0) \geq 0 \iff \inf\{f(z)-g(z):z\in D\} \geq 0$$

$$\iff f(z) \geq g(z) \iff f \geq g .$$

From the above relationships, the erosion $\Phi_f(f) = f\ominus g^S$ has the kernel:

$$\xi(g) = \{f \in \vartheta : f \geq g\} \ . \qquad (3\text{-}14)$$

In a manner similar to the set-processing system, let a function-processing system, Φ_f, be increasing. Then $g \in \kappa(\Phi_f)$ and $f \geq g$ imply $f \in \kappa(\Phi_f)$. Thus, $g \in \kappa(\Phi_f) \rightarrow \xi(g) \in \kappa(\Phi_f)$ from Equation (3-14). Hence, $\kappa(\Phi_f)$ is the union of $\xi(g)$ for all $g \in \kappa(\Phi_f)$, and thus the union of kernels $\xi(g)$ corresponds to the supremum of erosion by g. Therefore, any TI increasing system can be represented using mathematical morphology, as expressed in the following theorem:

Theorem 3.1 [10]: Any translation invariant and increasing function-processing system defined as a class of u.s.c. functions closed under translation can be represented exactly as the supremum of erosion by all its kernel functions, i.e.,

$$[\Phi_f(f)](x) = \sup_{g \in \kappa(\Phi_f)} \{(f \ominus g^S)(x)\} \ . \qquad (3\text{-}15)$$

3.3.2 Basis Representation of Increasing TI System

Since the kernel of an increasing TI system has an infinite number of elements, and an infinite number of erosions are needed to implement this system, theorem 3.1 is of no practical importance. Therefore, Maragos [10] introduced a concept of so-called basis of system,

which he defined as the collection of minimal kernel elements. The central idea of this approach can be extracted from the following example [10]. Consider a very simple increasing TI system Φ: the set erosion by B. Suppose that it is not originally known that $\Phi(X) = X \ominus B^s$, but only its kernel is known, $\kappa = \{A : A \supseteq B\}$. Since every subset A of E is such that $A \supseteq B$ belongs to κ, the kernel of Φ has an infinite number of elements. Equation (3-11) states that

$$\Phi(X) = \bigcup_{A \in \kappa} X \ominus A^s \text{ for all } X,$$

and, hence Φ is realized as an infinite number of erosions. The erosion $X \ominus A^s$ is decreasing with respect to A. Hence, $X \ominus A^s \subseteq X \ominus B^s$ for all $A \in \kappa$ and the infinite union of the erosions,

$$\bigcup_{A \in \kappa} X \ominus A^s ,$$

is equal to just one erosion, i.e., the erosion $X \ominus B^s$ by the element B of the kernel κ. Thus, it is possible to revert to the original unredundant definition of the system by realizing it with only one of its kernel elements.

In the general case, the kernel, $\kappa(\Phi)$, of a set-processing system, Φ, equipped with the ordering relations of set inclusion, is a partially ordered set. A kernel element is minimal in $(\kappa(\Phi), \subseteq)$ if and only if it is not preceded (with respect to $\subseteq$) by any other kernel element. If the system Φ is also increasing, and M is

one of its kernel elements, then $\xi(M) \subseteq \kappa(\Phi)$. Thus, any set $A \supseteq M$ belongs to $\kappa(\Phi)$, and knowing M is equivalent to knowing $\xi(M)$. In addition, $X \ominus A^S \subseteq X \ominus M^S$ for any input set X. Thus, in representing the system Φ as a union of erosion, the erosion by M contains the erosion by any other set in $\xi(M)$, and it is the only one needed. The collection of minimal kernel elements (sets) of Φ is henceforth defined as the basis of Φ, denoted as:

$$\beta(\Phi) = \{M \in \kappa(\Phi): A \in \kappa(\Phi)$$

and

$$A \subseteq M \rightarrow A = M\} \ . \tag{3-16}$$

Similarly, the basis of function-processing system Φ_f can be defined as follows:

$$\beta(\Phi_f) = \{g \in \kappa(\Phi_f): f \in \kappa(\Phi_f)$$

and

$$f \leq g \rightarrow f = g\} \ . \tag{3-17}$$

Maragos [10] proved that the basis of an increasing TI system exists if the system is also upper semicontinuous (u.s.c.). According to Matheron [1], dilation, erosion, opening and closing are u.s.c. mappings. Moreover, any finite union or intersection of u.s.c. TI set-processing systems is an u.s.c. system. The same is true for function-processing systems if the finite union is replaced with maximum and intersection with minimum.

Therefore the increasing and u.s.c. TI set-processing system Φ can be represented as the union of erosions by its basis elements, i.e., for any $X \in \mathcal{P}$,

$$\Phi(X) = \bigcup_{M \in \beta(\Phi)} X \ominus M^s \quad . \tag{3-18}$$

Theorem 3.2 [10]: Similarly, increasing and u.s.c. TI function-processing system Φ_f is exactly represented as the supremum of erosions by all of its basis functions, i.e., for any $f \in \mathcal{P}$ and any $x \in D$,

$$[\Phi_f(f)](x) = \sup_{g \in \beta(\Phi_f)} \{(f \ominus g^s)(x)\} \quad . \tag{3-19}$$

3.3.3 Linear Shift-Invariant System Representation

A linear shift-invariant (LSI) system is viewed as an FP filter that commutes only with a shift with respect to the argument of its input functions. From Equation (3-6), since the TI system commutes with a translation which is a shift with respect to both the argument and amplitude, the LTI system is an LSI system whose dc-gain is equal to one. Let $h(x)$, $x \in D$, denote the impulse response of LSI system Φ_l. If

$$\int_D h(x)dx = 1$$

(or

$$\sum_{n \in D} h(n) = 1),$$

LSI system Φ_l can be an LTI system. Maragos [10] provided the following theorem with proof concerning the increasing property requirement.

Theorem 3.3 [10]: A linear shift-invariant system is increasing if and only if its impulse response is non-negative everywhere.

Then we can make a kernel representation for a linear shift-invariant system, using the theorems 3.1 and 3.3:

Theorem 3.4 [10]: Let Φ_l be a linear shift-invariant system defined as u.s.c. functions closed under translation. Also let its impulse response, h(x), satisfy the following two conditions:

1) $h(x) \geq 0$ for all $a \in D = R^m$ (or Z^m), and

2) $\int_D h(x)dx = 1$ (or $\sum_{n=D} h(n) = 1$).

Then Φ_l is exactly represented as the supremum of erosions by all its kernel functions, $g \in \kappa(\Phi_l)$; thus, for any f and $x \in D$,

$$(h*f)(x) = \sup_{g \in (\Phi_l)} [\inf_{z \in D} \{f(z)-g(z-x)\}] \ . \qquad (3\text{-}20)$$

For discrete LTI systems, a sufficient condition to finding a non-empty basis in their kernel is to have an impulse response of finite extent. Such a system can be represented as the supremum of erosions by all its basis functions, as given in the following theorem:

Theorem 3.5 [10]: Let h(n), $n \in Z^m$, be the finite-extent impulse response of an increasing LTI discrete system, Φ_l, which is defined as a class, φ, of real-valued sampled functions

closed under translation. Then, the basis of Φ_1 is equal to

$$\beta(\Phi_1) = \{g\in\varphi: \sum_{k\in Ros(h)} h(k)g(-K) = 0 \text{ and } g(n) = -\infty \iff h(-n) = 0\} . \quad (3\text{-}21)$$

Furthermore, Φ_1 can be represented exactly as the supremum of erosions by all of its basis functions $g\in\beta(\Phi_1)$. That is, for any $f\in\varphi$ and $n\in Z^m$,

$$(h*f)(n) = \sum_{k\in Ros(h)} h(k)f(n-k)$$

$$= \sup_{g\in\beta(\Phi_1)} [\min_{k\in[Ros(g)]n} \{f(k) - g(k-n)\}] . \quad (3\text{-}22)$$

The proofs, used to find an implementing algorithm in Chapter IV, are also provided by Maragos [10].

1) Basis: Call β the class of functions given by Equation (3-21), and let m be the true basis of Φ_1. We must show that $m = \beta$; β is nonempty, because $g^*\in\beta$, where $g^*(n)= 0$ iff $h(-n) \neq 0$ and $g^*(n) = -\infty$ otherwise. Let now $g\in\beta$, then $g\in\kappa(\Phi_1)$ because $h*g(0) = 0$. Is g minimal? Suppose it is not. Then there is $f\in\kappa(\Phi_1)$, such that $f \leq g$ and $f \neq g$. Since $h(n) \geq 0$ for all n, $0 \leq h*f(0) \leq h*g(0) = 0 \iff h*f(0) = 0$. Since $f \leq g$ and $g(n) = -\infty$ for all $n\notin Ros(g)$, there exists $k\in Ros(g)$ such that $f(k) < g(k)$; this implies $h(-k)f(k) < h(-k)g(k)$ and thus $h*f(0) < h*g(0) = 0$: Contradiction! Hence, $g\in m$, and thus, $\beta \subseteq m$.

Let now $g \in m$. All the basis functions g must have a minimal region of support $G = [Ros(h)]^S$, because only the indexes $n \in G$ are required for $g(n)$ in computing $h*g(0)$. Thus, g satisfies the second of the two requirements of Equation (3-21). Suppose that $g \notin \beta$, then $h*g(0) = p > 0$, and consider the function $f \in \varphi$ with $f(n) = g(n)-p$, $n \in G$. Then, $f \leq g$ and $f \neq g$. However, $h*f(0) = h*g(0)-p = 0$, and hence f is a kernel function of Φ_l that precedes g. Hence, g is not a minimal element: Contradiction! Therefore, $g \in \beta$, and thus $m \subseteq \beta \subseteq m \Rightarrow m = \beta$.

2) Representation: Since Φ_l is translation-invariant, increasing, and u.s.c. (due to the finite extent of h), it can be represented exactly as the supremum of erosions by its basis functions according to Theorem 3.4.

An alternative proof proceeds as follows: Let $Ros(h) = \{k_1, k_2, \ldots, k_N\}$ be the N-point finite region of support of $h(n)$. Let also $h_i = h(k_i)$, $f(n-k_i) = f_i$, and $g(-k_i) = g_i$, with $i=1,2,\ldots,N$ and $k_i \in Ros(h)$. Then, it must be proved that for all n,

$$\sum_i h_i f_i = \sup_{g \in \beta(\Phi_l)} [\min\{f_i - g_i\}] , \qquad (3\text{-}23)$$

subject to

$$\sum_i h_i g_i = 0, \quad \sum_i h_i = 1, \quad h_i > 0.$$

For any n, all f_i and $\Sigma_i h_i f_i$ are arbitrary but fixed real numbers. Hence, among all functions in $\beta(\Phi_1)$ a basis function g^* defined by $g_i^* = g^*(-k_i) = f_i - \Sigma_i h_i f_i$, $i = 1,2,\ldots,N$ may always be found. For each $g \in (\Phi_1)$, there is a $j \leq N$ such that

$$f_j - g_j \leq f_i - g_i \text{ for all } i ,$$

$$\Rightarrow h_i(f_j - g_j) \leq h_i(f_i - g_i) \text{ for all } i ,$$

and

$$\Rightarrow \Sigma_i h_i(f_j - g_j) \leq \Sigma_i h_i(f_i - g_i) \text{ for all } i . \qquad (3\text{-}24)$$

Since $\Sigma_i h_i = 1$ and $\Sigma_i h_i g_i = 0$,

$$f_j - g_j \leq \Sigma_i h_i f_i . \qquad (3\text{-}25)$$

From Equations (3-24) and (3-25), if the following lower and upper bound of inequality are chosen,

$$g_j = f_j - \Sigma_i h_i f_i , \qquad (3\text{-}26)$$

and

$$g_i = g_j + f_i - f_j \text{ for all } i \neq j , \qquad (3\text{-}27)$$

then the result is $\sup_g [\min_i \{f_i - g_i\}] = \sup_g \{f_j - g_j\}$ $= \Sigma_i h_i f_i$. Thus the proof is complete.

3.3.4 Summary

In summary, a simple example, including the previously discussed main concepts and theorems, is presented as follows:.

Consider the one-dimensional, three-point, low pass-filter, Φ_1, whose impulse response is

$$h(n) = a\delta(n-1) + b\delta(n) + a\delta(n+1),$$

where $2a + b = 1$ and $a,b > 0$. Since $a,b > 0$ and $2a + b = 1$, this filter is increasing and translation invariant. From Equation (3-13), its kernel is equal to

$$\kappa = \{f: af(n-1) + bf(n) + af(n-1) \geq 0\} .$$

From Equation (3-22), its minimal elements (basis) are the function g such that

$$\begin{aligned} & h(-1)g(1) + h(0)g(0) + h(1)g(-1) \\ & \quad = ag(1) + bg(0) + ag(-1) \\ & \quad = 0 . \end{aligned}$$

Hence, the structuring functions $g(n)$ can be chosen,

$$g(-1) = r \in R,\ g(0) = s \in S,\ g(1) = (-ar-bs)/a$$

and

$$g(n) = -\infty \text{ if } n \notin \{-1,0,1\} .$$

Thus, the output of system Φ_1 can be written from Equation (3-22),

$$\begin{aligned} (h*f)(n) &= af(n-1) + bf(n) + af(n+1) \\ &= \sup_{(r,s) \in R^2} [\min\{f(n-1) - r, f(n) - s, f(n+1) \\ &\qquad + (ar+bs)/a\}] . \end{aligned}$$

IV. IMPLEMENTATION OF MORPHOLOGICAL FILTERS IN TYPICAL PARALLEL ARCHITECTURES

A variety of image processing tasks can be performed by combining basic morphological filters, e.g., erosions, dilations, openings, and closings. Applications for morphological filters were discussed section 3.2 and in this chapter, selected typical parallel architectures, those in such categories of operator parallelism as pipelining, are discussed. In addition, the erosion algorithms for SP, FSP ,and FP cases are discussed. Because dilation is essentially complementary to the erosion of the original image, and of the same computational complexity, and opening and closing are combinations of erosion and dilation, the point of concentration is placed upon erosion.

Although the morphological representation of linear shift invariant filters was discussed in section 3.3, using minimal kernel elements (as the basis), the realization of such representation remains impractical because of infinite numbers of basis functions. However, suppose the use of a special parallel architecture which cannot support multiplying operations. Linear filtering then becomes necessary. In section 4.4,

a new architecture and corresponding parallel algorithm to implement a linear filter (convolution) will be presented to remedy this situation. Moreover, the reason why morphological representation of LSI filtering is impractical is discussed.

4.1 Considerations on Effective Architecture for the Implementation of Morphological Filters

Since morphological transformation locally modifies geometric features of images by introducing the concept of structuring elements, this transformation is a type of neighborhood operation. Neighborhood operations, which are nearly equivalent to what Preston et al. [21] call cellular logic operations, are illustrated in Fig. 4.1.

In Fig. 4.1, a program step defines an operation both in terms of which pixels in the input data contribute to an output value and in terms of the parameters and constants involved. Each operation on a pixel and its defined neighborhood shall be repeated at each pixel in the array , in each case using the value of pixels at time t to compute the new value of each pixel at time $t+\delta t$. Thus if a defined neighborhood of for a two-dimensional operation is the $M \times M$ region surrounding each pixel A_{xy}, then the new value of each pixel is given by C_{xy} where

$$C_{xy} = f(A_{jk})$$

$$x-1/2M \leq j \leq x + 1/2M$$

$$y-1/2M \leq k \leq y + 1/2M \qquad (4-1)$$

for all x and y in the array.

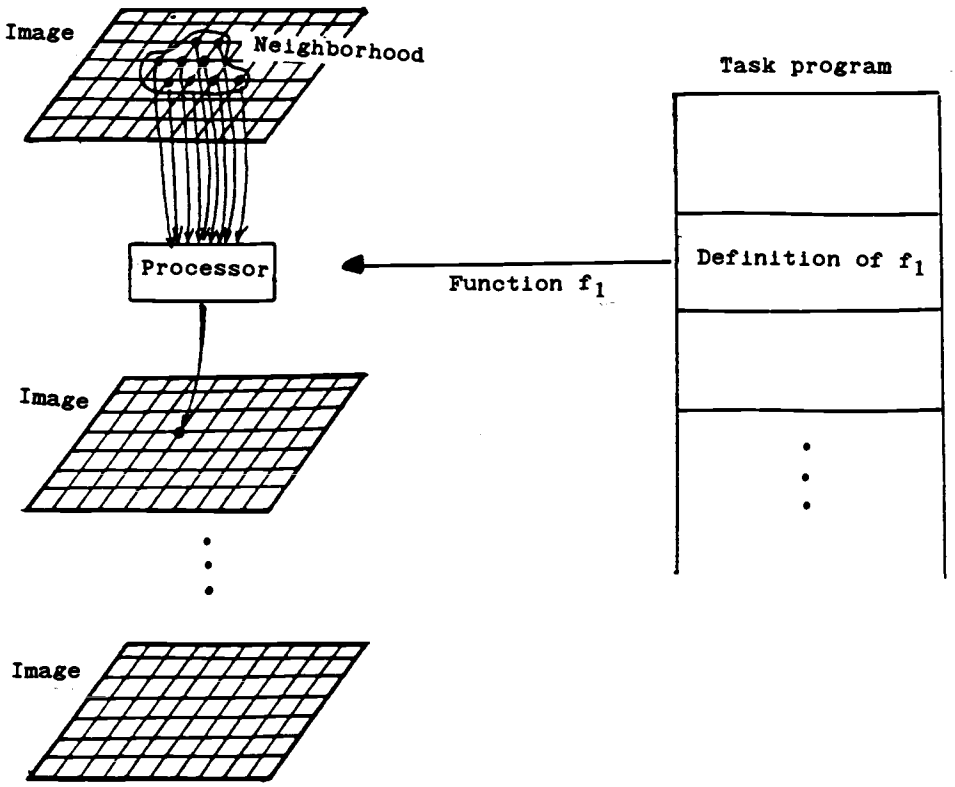


Figure 4-1 Neighborhood operation.

Unfortunately, neighborhood operations performed by general purpose processors (serial processors) are extremely slow. This limitation results from such architectural factors as memory structure (a one-dimensional vector vs the two dimensional array data structure of an image), computational power (one central processing element which must perform all program/data access and computations), and limited address space (a typical image is often too large to reside in

main memory storage). Therefore, various special purpose architectures have been proposed to surmount these problems.

4.1.1 Parallel Array Processor

Figure 4.2 indicates a two-dimensional parallel array architecture with a connection pattern for a 3 × 3 window configuration. Each element of the array is a distinct processor element, containing a memory register for holding the current pixel value. The two-dimensional configuration of pixels constitutes a digitized image. Each processing element is directly connected to the set of nearest-neighbor elements in the array, forming the neighborhood relationship.

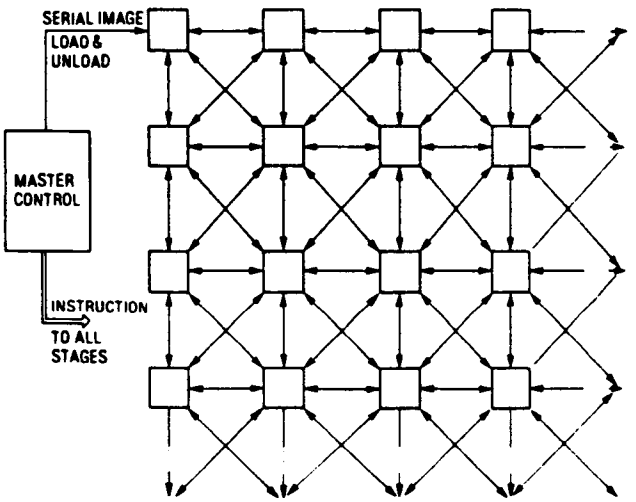


Figure 4.2 Parallel array processor.

Processing elements of the parallel array contain a neighborhood-transform logic module for computing

pixel transitions. Transition instructions are broadcast in parallel to each logic module by the master control, which then stores the complete sequence of transition instructions constituting the algorithm. Each pixel transition is effected simultaneously on all pixels. The output of each neighborhood logic module is a new pixel state, replacing the old values in the pixel state register. A computer image transformation algorithm is executed by a sequence of logic module program steps and pixel state neighborhood transformations.

Although digitized image dimensions may often exceed 1000 x 1000 pixels in many applications, the largest arrays produced to date are only on the order of 256 x 256 [34]. Large images must be partitioned into image segments and each segment is processed in turn. Furthermore, segment border effects propagate into the segment when multiple neighborhood transformations are applied, necessitating extremely costly and time-consuming I/O hardware and software subsystems for rapid segment swapping.

4.1.2 Pipeline Image Processor

The concept of pipeline processing has been developed to match image processing system architecture to serial data inputs. At each stage, a limited amount of buffer memory and only one processor are available.

Typically, only a limited number of consecutive lines of the input data are stored in the buffer memory. The processor obtains data from this buffer, computing the output data in a serial fashion. But at the same time, the second stage in the pipeline is already busy with the next step in the operation, delivering data for the third stage long before the entire first stage image is completed.

Although pipelining is a concept implemented in many architectures, none employ it to a greater extent than the cytocomputer developed at the Environmental Research Institute of Michigan (ERIM), where Sternberg [27] was the main investigator. The inspiration for the development of this alternative structure was the practical shortcomings of parallel array processors. Lougheed and McCubbrey [26] have compared cytocomputer and parallel array processor architectures, showing that the cytocomputer offers several advantages with respect to array architecture, including low complexity, high bandwidth, and considerable architectural flexibility.

Therefore, pipeline image processor (cytocomputer) designs are clearly preferable to parallel arrays in terms of cost effectiveness, processing capability, and architectural flexibility in real-world situations. Furthermore, the main objective of the cytocomputer is to support morphological operations. In this sense,

this chapter focuses upon on cytocomputer architecture, including discussion of algorithms for morphological operations and LSI filter implementation for this system.

4.2 Overview of Cytocomputer Architecture

The name "cytocomputer" is derived from *cyto*, the Greek word for cell, referring to its cellular architecture, and "computer" because a cytocomputer is a computer in the formal sense of the word. First proposed by Sternberg [27] in 1976, a cytocomputer (Fig. 4.3) consists of a serial pipeline of neighborhood processing stages with a common clock, in which each stage in the pipeline performs a single transformation of an entire image. Pictures are entered into the pipeline as a stream of pixels in sequential line-scanned format, progressing through the pipeline of processing stages at a constant rate. Following initial latency in filling the pipeline, processed images are produced at the same rate at which they can be entered. Shift registers within each stage store two contiguous scan lines, while window registers hold the 9 neighborhood pixels which constitute the 3 × 3 window input of a neighborhood logic module. This module performs a programmed transformation of the center pixel based on the

values of the center and its 8 neighbors. Neighborhood logic transformations are

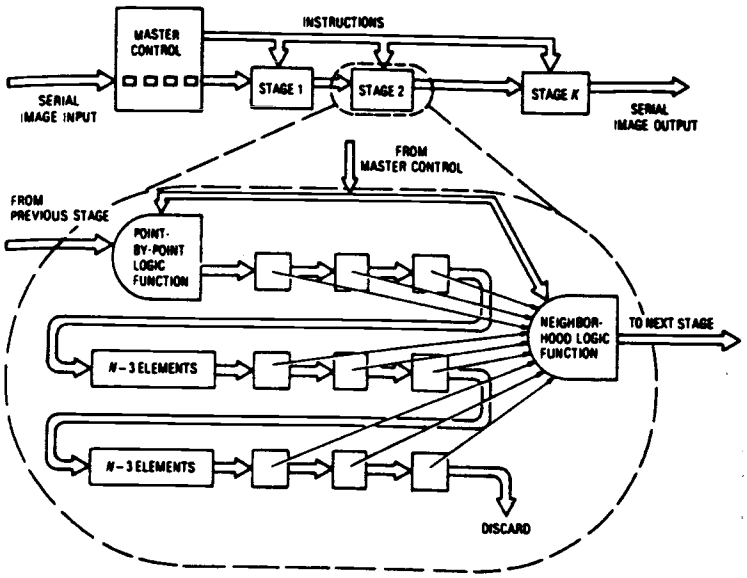


Figure 4.3 Cytocomputer block diagram.

computed within the data transfer clock period, allowing the output of a stage to appear at the same rate as its input. At each discrete time interval, a new pixel is clocked into the stage. Simultaneously, the contents of all delay units are shifted one element. In addition, operations which do not involve the states of a pixel's neighbors, such as scaling or bit setting, can be performed in a separate point-by-point logic section to simplify the neighborhood logic circuit.

An effort to visualize the transformation process is presented as a 3 x 3 window in Figure 4.4. The processing stage storage section shows the contents of the latches after pixel $A_{6,6}$ has been read. The contents of the neighborhood latches allow the stage to compute

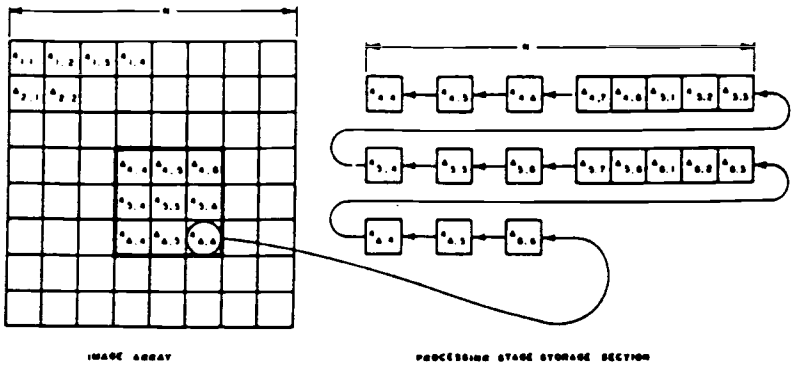


Figure 4.4 Moving window implemented with shift register storage.

the transformed value of the pixel in the center latch, $A_{5,5}$. This transformed pixel becomes an element of the serial input to the next stage. From this example, it may be seen that the latency of a stage is equal to N + 2 time steps, N being the line length. Processing occurs as a series of 3 x 3 windows, following each other across the image as each one processes the output of the previous stage.

Figure 4.5 shows the ERIM image processing laboratory. It contains two different stage types: two-dimensional and three-dimensional transformation stages, of which there are, respectively, 88 and 25.

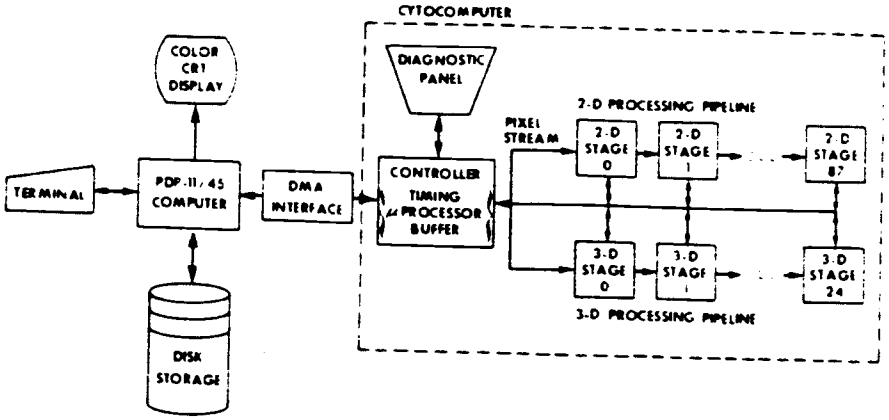


Figure 4.5. ERIM image processing laboratory.

4.3 Examples of Morphological Transformation Algorithm

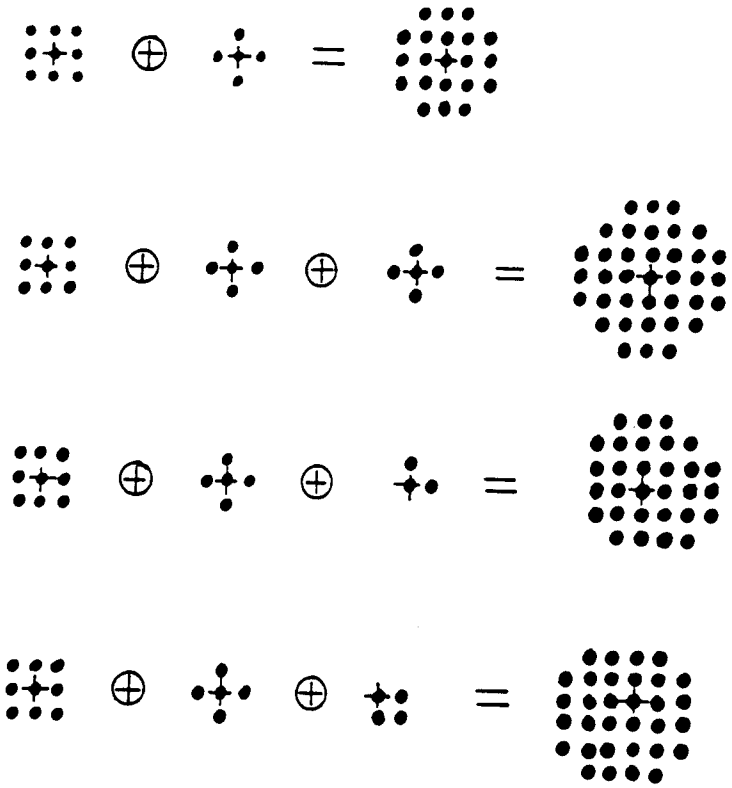
4.3.1 Set-Processing Erosion Algorithm

Set-processing erosion can be executed by 'ANDing' all binary pixels within designed structuring element. Figure 4.6 shows various possible structuring elements using 3 × 3 windows. According to the properties of dilation and erosion (section 2.1.1),

$$X\ominus(B_1\oplus B_2)=X\ominus B_1\ominus B_2 \ .$$

Therefore, an input image X can be eroded by larger structuring elements by successively eroding structuring elements B_1 and B_2, which can be represented in a 3 × 3 window.

The following procedure describes set-processing erosion steps for a two-dimensional stage (Figure 4.7):



(• = object points, ⊕ = origin)

Figure 4.6 Large structuring element formation as the dilation (Minkowski sum) of simpler sets (within 3 × 3 window).

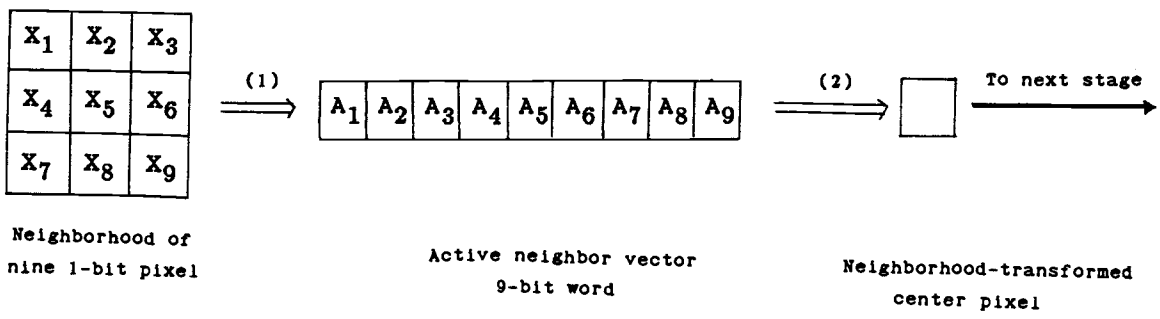


Figure 4.7 Set-processing erosion.

1) Mapping of the 9 neighborhood pixels, X_n (n=1,...,9), into an active neighborhood vector:

$$A_n = X_n \quad (n=1,\ldots,9)$$

2) Using the 9-bit active neighborhood vector as index, obtain a neighborhood-transformed center pixel from a table of 512, 1-bit values preprogrammed as follows:
   a. Table index = Active neighborhood vector; and
   b. Make all required tables by "ANDing" all bits within the structuring element, which can also be represented by a 9-bit word.

4.3.2. Function-Processing Erosion Algorithm

Arithmetic representation of function processing erosion $f \ominus g$ (Equation 3-4b) can be realized directly in a three-dimensional stage with the following procedure (Fig. 4.8):

1) Calculate Equation (3-4b) and scale, using predetermined appropriate bias values ensuring the positive values as follows:

$$Q_n = \text{Bias} + P_n - G_n \ ,$$

   where the additions are modulo 256, and G_n is n^{th} cell of the structuring function;
2) Obtain the min $\{Q_1,\ldots,Q_n\}$; and
3) Subtract the bias from the result of step 2.

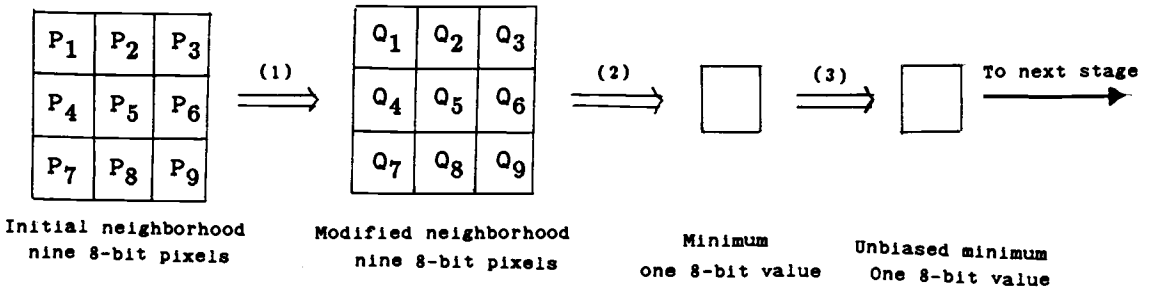


Figure 4.8 Function-processing erosion.

Function-set-processing erosion is a special case of function-processing erosion, i.e., (G_n = 0, Bias = 0).

Morphological transformations of gray-scale images by a gray-scale structuring function is understood in terms of the gray-scale image umbra. In accordance with the umbra concept, two-dimensional gray-scale images can be interpreted as three-dimensional sets. Similar to the set-processing structuring element, larger structuring element can be constructed to dilate subsets of 3 x 3 x 3 windows. For practical purposes, spherical structural elements are used in rolling ball opening algorithms. Sternberg [4] created a sphere (i.e., a digital ball) by dilating a single point with a sequence of 26 gray-scale neighborhood operations. However, there is no general algorithm for the construction of a particular structuring element by sequential dilation of 3 x 3 x 3 window subsets. This problem and related image analysis are suitable topics for further study.

4.4 Linear-Shift-Invariant Filtering Algorithm

In section 3.3, morphological representation (Theorem 3.5) of a linear-shift-invariant system was obtained. Maragos [10] noted that if the amplitude of basis functions is quantified and their range is bound between certain limits, the true response of the LSI filter can be approximated. In addition, a linear filter can be realized only by using max-min and algebraic additions, thus avoiding algebraic multiplications. However, there is some disagreement with these comments since Maragos omitted the following facts which cause Equations (3-21) and (3-22) to be impractical for use with existing image processors:

1) To obtain basis functions, $g(k)$, from Equation (3-21) without the use of multiplications, an N-1 dimensional memory array, where N is the number of impulse response terms, is required. When 3×3 window and 2^8 gray-level quantification is considered, there should be a $256^{(9-1)}$ look-up table array.
2) Moreover, to realize Equation (3-22), the $f(k) - g(k-n)$ term must be calculated $256^{(9-1)}$ times.

However convenient this algorithm may be, it is impractical. The reader may ask, then, why the morphological

representation of a linear shift-invariant filter may be derived? The answer is as follows:

1) Although morphological filters are nonlinear transformations, linear-shift-invariant filters can be represented by nonlinear filters; and
2) The basis (or kernel) function is uniquely determined by Equations (3-26) and (3-27), from which a convolution algorithm may be derived while using morphological architecture.

Most conventional image enhancement techniques use a linear spatial filtering method, e.g., low pass filtering for noise suppression and/or image smoothing, and high pass filtering for image sharpening. The following discussion presents an approach to determination of an algorithm supporting linear spatial filtering (i.e., a convolution algorithm) using morphological architecture. Of necessity, this method submits to abandonment of some of the advantages of linear filtering due to limited window size (i.e., 3 x 3 in the case of the cytocomputer). Nevertheless, the following algorithm extends the power of the cytocomputer considerably.

Consider the following linear convolution,

$$Q(n,m) = F*H = H*F = \sum_{k=-\infty}^{\infty} \sum_{r=-\infty}^{\infty} h(k,r)f(n-k,m-r)$$

where F = input image,

Q = output image (filtered image),

H = impulse response (mask or moving window),

f(.,.) = gray scale value of the input pixel, and

h(.,.) = impulse response at (.,.).

Pratt [17] lists several noise cleaning masks and edge sharpening masks as follows:

1) Noise-cleaning mask (low-pass form).

$$\text{Mask1.}\quad H = 1/9 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\text{Mask2.}\quad H = 1/10 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\text{Mask3.}\quad H = 1/16 \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

2) Edge sharpening mask (high-pass form).

$$\text{Mask1.}\quad H = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$\text{Mask2.}\quad H = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$\text{Mask3.}\quad H = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

As may be seen, F is symmetric. Therefore, a 3 x 3 moving window, H, may be written as follows, without loss of generality:

$$H = \begin{bmatrix} h_2 & h_1 & h_2 \\ h_1 & h_0 & h_1 \\ h_2 & h_1 & h_2 \end{bmatrix} , \qquad (4-2)$$

where $\sum \sum h = h_0 + 4h_1 + 4h_2 = 1$. (4-3)

From Equation (3-26),

$$g(0,0) = f(n,m) - \sum_{i=1}^{3} \sum_{j=1}^{3} h(i,j) f(n-i,m-j) . \qquad (4-4)$$

Using Equation (4-3),

$$f(n,m) = f(n,m)(h_0+4h_1+4h_2) .$$

Then Equation (4-4) can be written,

$$\begin{aligned} g(0,0) = {} & [\{f(n,m) - f(n+1,m)\} + \{f(n,m) \\ & - f(n,m+1)\} \\ & + \{f(n,m) - f(n,m-1)\} + \{f(n,m) \\ & - f(n-1,m)\}]h_1 \\ & + [\{f(n,m) - f(n+1,m+1) + \{f(n,m) \\ & - f(n+1,m-1)\} \\ & + \{f(n,m) - f(n-1,m+1)\} + f(n,m) \\ & - f(n-1,m-1)\}]h_2 . \end{aligned} \qquad (4-5)$$

Since Equations (3-26) and (3-27) are true for all i and j ($i \neq j$), from Equation (3-23) the resultant center pixel is

$$q(n,m) = f(n,m) - g(0,0) . \qquad (4-6)$$

The above relations (4-5) and (4-6) provide a parallel algorithm which can then be implemented on the cytocomputer with only slight architectural modifications.

As shown in Figure 4.9, to synchronize (execute within one clock) with other stages in the pipeline, stages K and K+1 are divided. The parallel connected stages K and K' have such architectures that are similar to three-dimensional stages (see Fig. 4.5), with the exception of the bit length of registers A_1 and A_2. With this architecture, linear filters can be realized by means of the following algorithm:

0) Input an 8-bit pixel from the previous stage in the pipeline;
1) Calculate the terms within brace ({ }) in Equation (4-5);
2) Add M_2, M_4, M_6, and M_8 and add M_1, M_3, M_7, and M_9;
3) Find a value from a look-up table preprogrammed to provide A_1h_1 and A_2h_2;
4) Add the two values, B_1 and B_2;
5) Subtract the result of step 4 from f_5;
6) Find a value from a table preprogrammed to provide denormalization (in case of $\sum \sum h \neq 1$); and
7) Output an 8-bit pixel to the next stage.

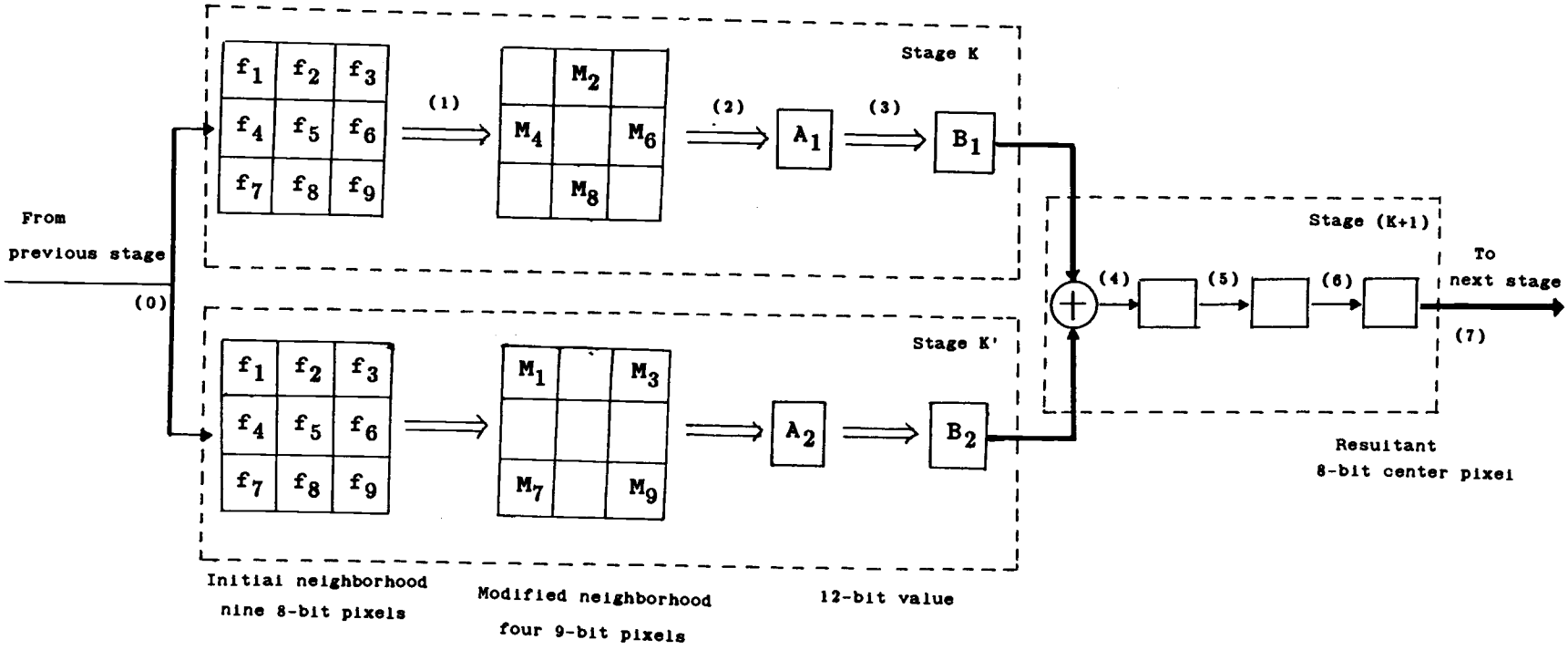


Figure 4.9 Convolver using 3 x 3 moving window.

4.5 Summary

In this section we suggested a new architecture and presented a corresponding algorithm in order to realize linear shift-invariant filters as well as morphological filters. Using the symmetrical nature of the most commonly used linear filtering masks, a simple algorithm was derived for implementation on systems based upon pipeline parallel architecture. Preserving both the pipeline architecture and the concept of stages used in the morphological filter realization, real time (TV rate) image analysis can be maintained with the use of this algorithm. Therefore, the material provided in this study may be used for the considerable improvement of the image analyzing capabilities of mathematical morphology, which in its original form is non-linear.

A number of areas remain, however, as appropriate topics for future study. The principal remaining problems are (1) to determine an architecture and an algorithm suitable for use in the case of non-symmetric windows, (2) to determine an architecture and an algorithm which can preserve real time processing in the case of window sizes larger than 3×3, and (3) to conduct an analytical study of hybrid-filters, which can

be constructed by combining linear and morphological filters.

V. CONCLUSIONS

5.1 Conclusions and Contributions of this Study

5.1.1 Analysis of Mathematical Morphology

Since the term "mathematical morphology" has not to date become a popular genre in image processing or in system theory, the concepts of mathematical morphology, including the relationship between sets and functions, the various classes of morphological transformations and applications, and its relative ease of readability, were discussed and analyzed. To introduce these concepts was one of the original goals of this study.

5.1.2 Relationship Between Theory and Realization

The theoretical background of mathematical morphology was provided by Matheron [1] and Serra [2], and the umbra concept was added by Sternberg [4]. Moreover, Sternberg developed a special architecture, the pipeline image processor, which operates very well in conformity with the principles of mathematical morphology.

In this study, morphological transformations as neighborhood operations, which are feasible on existing

parallel architecture, were discussed. Among the number of proposed special purpose parallel architectures capable of realizing neighborhood operations, the most effective architecture, based upon reasonable criteria of effectiveness, was selected . This study then provided a morphological transformation algorithm, in accordance with the classification of signals and system, for the selected architecture.

5.1.3 Linear Filter Realization

In this study, a new architecture and corresponding algorithem, capable of realizing linear shift-invariant filters as well as morphological filters, has been presented. Since most conventional image enhancement techniques use linear spatial filtering method, linear filtering (convolution) cannot be avoided. Although Maragos [10] provided a morphological representation of a linear shift-invariant system, it has proved to be impractical due to excessive memory size and calculation time requirements. Furthermore, the architecture selected in the study cannot execute linear filtering algorithms in its existing form due to the structure of its mathematical morphological orientation.

Consequently, the architecture suggested in study, with the algorithm presented, will provide considerable improvement in the power of image processing systems,

as well as improvements in the application fields of mathematical morphology.

5.2 Suggestions for Future Study

The universe of all possible object shapes is vast. There is, therefore, a huge number of potential structuring elements. The systematic analysis of structuring elements would prove an interesting and useful topic for future research. Specifically, if arbitrary structuring functions capable of dilating subsets of 3 x 3 x 3 windows could be constructed, it would constitute a considerable extension of the applications of function processing filtering.

Most morphological filters are combinations of erosion and dilation, each with its own filtering characteristics. A new filter could be constructed by combining erosion and dilation in a different manner, but its construction should be accompanied by filtering performance analysis of the new filter and its related fields of application.

The research field for new applications of morphological filters in image and signal processing is open for additional attention, especially in such problems as the extraction of peaks and valleys from signals, currently requiring geometrical formulations. Previous research has mainly concentrated on image analysis. It

may now be worthwhile to extend applications knowledge to the fields of any two-dimensional signal manipulation (e.g., image data-base management).

BIBLIOGRAPHY

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

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

[3] S. R. Sternberg, "Biomedical Image Processing," *IEEE Computer* ,Jan. 1983 ,pp. 22-34.

[4] S. R. Sternberg, "Greyscale morphology," *Comput. Vis. Graph. Image Processing*, vol. 35, 1986, pp. 333-355.

[5] S. R. Sternberg, "Parallel Architectures for Image Processing," in *Real Time/Parallel Computing Image Analysis* (Onoe, Preston, and Rosenfeld, eds.), Plenum Press, New York, 1981.

[6] 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, Aug. 1978, pp. 632-635.

[7] V. Goetcherian, "From binary to grey tone image processing using fuzzy logic concepts," *Pattern Recog.*, vol. 12, 1980, pp. 7-15.

[8] C. Lantuejoul and J. Serra, "M-filter," in *Proc. 1982 IEEE Intern. Conf. on Acoust., Speech., and Signal Processing*, Paris, France, May 1982, pp. 2063-2066.

[9] M. Werman and S. Peleg, "Min-max operators in texture analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-7, no. 6, Nov. 1985, pp. 730-733.

[10] P. Maragos, *A unified theory of translation-invariant systems with applications to morphological analysis and coding of images*, Unpublished doctoral dissertation, School Elec. Eng., Georgia Inst. Technol., Tech. Rep. DSPL-85-1, Atlanta, GA, July 1985..

[11] P. Maragos and R. W. Schafer, "Morphological filters-Part I: Their Set-Theoretic Analysis and Relations to Linear Shift-Invariant Filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, Aug. 1987, pp. 1153-1169.

[12] P. Maragos and R. W. Schafer, "Morphological filters-Part II: Their relations to median, order-statistic, and stack filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, Aug. 1987, pp. 1170-1184.

[13] P. Maragos, "Tutorial on advances in morphological image processing and analysis," in *Proc. SPIE, Vis. Commun. Image Processing*, vol. 707, 1986, pp. 64-74.

[14] P. Maragos and R. W. Schafer, "Morphological skeleton representation and coding of binary images," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, Oct. 1986.

[15] T. R. Crimmins and W. M. Brown, "Image algebra and automatic shape recognition," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-21, Jan. 1985, pp. 60-69.

[16] S. G. Tyan, "Median Filtering: Deterministic Properties," in *Two-Dimensional Digital Signal Processing II: Transforms and Median Filters* (T. S. Huang, ed.), Springer Verlag, New York,1981.

[17] W. K. Pratt, *Digital Image Processing*, J. Wiley & Sons, New York,1978.

[18] W. B. Green, *Digital Image Processing (A system approach)*, Van Nostrand Reinhold Company, New York, 1983.

[19] R. C. Gonzalez and P Wintz, *Digital Image Processing*, Addison-Wesley Publishing Company, 1977.

[20] A Rosenfeld and A. C. Kak, *Digital Picture Processing*, vols. 1 & 2, Academic Press, New York, 1982.

[21] K. Preston, Jr., J. B. Duff, S. Levialdi, P. E. Norgren, and J. I. Toriwaki, "Basis of cellular logic with some applications in medical image processing," *Proc. IEEE*, vol. 67, no. 5, May 1979, pp. 826-856.

[22] J. C. Klein and J. Serra,"The texture analyzer," *J. Microscopy*, vol. 95, pt. 2, Apr. 1972, pp. 349-356.

[23] R. M. Lougheed, D. L. McCubbrey, and S. R. Sternberg, "Cyto-computers: Architectures for parallel image processing," in *Proc. Workshop Picture Data Descr. Manag.*, Pacific Grove, CA, Aug. 1980.

[24] S. R. Sternberg, "Language and architecture for parallel image processing," in *Pattern Recognition in Practice* (E. S. Gelsema and L. N. Kanal, eds.), North-Holland Publishing Company, 1980.

[25] E. Cloud and W. Holsztynski, "Higher efficiency for parallel processors," *Professional Program Session Record 14/4, Computer Vision, SOUTHCON '84*, 1984.

[26] R. M. Lougheed and D. L. McCubbrey, "The Cytocomputer: a practical pipelined image processor," in *Proc. 7th Annual International Symposium on Computer Architecture*, May 1980.

[27] S. R. Sternberg,"Automatic image processor," U.S. Patent 4,167,728.

[28] P. E. Danielsson and S. Levialdi, "Computer architectures for pictorial information systems," *IEEE Computer*, Nov. 1981, pp. 53-67.

[29] G. Choquet, *Topology*, Academic Press, New York, 1966.

[30] K. Preston, Jr. and L. Uhr, ed., *Multicomputer and Image Processing (Algorithms and Programs)*, Academic Press, New York, 1982.

[31] M. Onoe, K. Preston, Jr. and A. Rosenfeld, ed., *Real-Time /Parallel Computing (Image Analysis)*, Plenum Press, New York, 1981.

[32] J. Kittler and J. B. Duff, ed., *Image Processing System Architectures*, Research Studies Press, England, 1985.

[33] J. B. Duff and S. Levialdi, ed., *Languages and Architectures for Image Processing*, Academic Press, New York, 1981.

[34] L. Uhr, ed., *Parallel Computer Vision*, Academic Press, Orlando, FL, 1987.

APPENDIX

Appendix

Table of Symbols and Notations

E	Euclidean space
FP	function processing
FSP	function-and-set processing
G(f)	graph of function f; $G(f)=\{(x,t)\in E: f(x)=t\}$
inf	infimum (lower bound)
max	maximum
min	minimum
m-D	m-dimensional
nB	$B\oplus B\oplus\cdots\oplus B$ (n times)
o	origin
R	set of real numbers (one-dimensional Euclidean space)
Ros	region of support
R^m	m-D Euclidean space
SP	set-processing
sup	supremum (upper bound)
TI	translation invariant
USC(D)	the class of upper semi-continuous on set D
u.s.c.	upper semi-continuous
$U(f)$	umbra of function f
X^B	closing of X by B; $X^B=(X\oplus B^s)\ominus B$
X_B	opening of X by B; $X_B=(X\ominus B^s)\oplus B$
X^c	complementary set of a set X

X_h translate of set X by h; $X_h = X \oplus \{h\}$

X^r reflected set of B with respect to horizontal plane

X^s symmetric set of X about the origin; $X^s = \{-x, x \in X\}$

$X_t(f)$ cross-section of function f at level t

X/α similar of X with scaling factor $1/\alpha$ ($\alpha > 0$); $X/\alpha = \{x : x\alpha \in X\}$

Z^m m-dimensional integer space

Greek

$\beta(\Phi)$ basis of the transformation Φ

ξ kernel of the erosion

$\kappa(\Phi)$ kernel of the transformation Φ

$\Phi(X)$ set transformed of X with respect to set transformation Φ

Φ_f function processing morphological transformation

Φ_l linear transformation

Φ_α transformation depending on positive parameter α

$\emptyset$ empty set

Π picture or function

$\rho(E)$ space of all subset of E

ϑ subcollect of $\rho(E)$

$\oplus$ Minkowski addition

$\ominus$ Minkowski subtraction