Chapter 1

Logical Image Operators

Edward R. Dougherty
Texas A&M University
College Station, Texas

Junior Barrera
University of Sao Paulo
Sao Paulo, Brazil

All digital image processing algorithms involve logical functions operating on vectors of logical variables, where the logical vectors represent binary encodings of image data. *Ipso facto*, characterization of image operators is naturally set in a logical framework, and this is especially straightforward in the case of binary image operators. The central tasks of any applied operator theory are analysis and synthesis of operators. Analysis concerns operator effects and synthesis the design of operators to perform desired tasks. Since digital images are modeled as discrete random sets, operator design involves synthesis governed by probabilistic criteria relating input and output image processes. The methodology is to find suitable operator representations, examine the manner in which these representations interact with the image probability structure to characterize optimal operators, and develop tools that translate them into efficient image processing procedures. The approach taken in the present chapter is to use examples (collections of input-output pairs of images) as the knowledge source for the representation formalism.

1.1 Boolean Functions

A binary-valued function $\psi(x_1, x_2, \ldots, x_n)$ of binary variables $x_1, x_2, \ldots, x_n$ is called a *Boolean function*. We will denote binary variables and Boolean functions by lower-case italic and Greek letters, respectively. As a logical function, ψ possesses a logical sum-of-products *disjunctive-normal-form* representation in terms of the n variables $x_1, x_2, \ldots, x_n$:

$$\psi(x_1, x_2, \ldots, x_n) = \sum_i x_1^{p(i,1)} x_2^{p(i,2)} \cdots x_n^{p(i,n)} \tag{1–1}$$

where the "sum" denotes OR, the "product" denotes AND, and $p(i,k)$ is either $'$ (prime) or null, depending on whether the variable is complemented or not complemented. There are at most 2^n products in the expansion, and each product is called a *minterm*. The representation can be (nonuniquely) reduced to a sum of products containing a minimal number of logic gates; that is,

$$\psi(x_1, x_2, \ldots, x_n) = \sum_i x_{i,1}^{p(i,1)} x_{i,2}^{p(i,2)} \cdots x_{i,n(i)}^{p(i,n(i))} \tag{1–2}$$

Disjunctive normal form will be used for operator design, with reduction being employed to cut the logic cost of implementation.

The truth table formulation of ψ corresponds directly to the disjunctive normal form of Eq. 1–1. ψ is defined by a 2^n-row truth table of n variables in which each string $t_1 t_2 \cdots t_n$ of 0s and 1s is assigned a binary value $\psi(t_1 t_2 \cdots t_n)$. The correspondence between Eq. 1–1 and the truth table is given by the following rule: the minterm $x_1^{p(i,1)} x_2^{p(i,2)} \cdots x_n^{p(i,n)}$ appears in the expansion of Eq. 1–1 if and only if there is a 1-valued string $t_1 t_2 \cdots t_n$ in the truth table with $p(i, j)$ null if $t_j = 1$ and $p(i, j) ='$ if $t_j = 0$.

The product set $\{0, 1\}^n$ is composed of binary n-vectors and is a finite lattice under the partial-order relation $(x_1, x_2, \ldots, x_n) \leqslant (y_1, y_2, \ldots, y_n)$ if and only if $x_j \leqslant y_j$ for $j = 1, 2, \ldots, n$. We denote vectors of binary variables by bold-face lower-case letters, such as $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$. For any $\mathcal{A} \subset \{0, 1\}^n$, the *upper set* of $\mathcal{A}$ is defined by

$$\mathcal{U}[\mathcal{A}] = \{\mathbf{y}\text{: there exists } \mathbf{x} \in \mathcal{A} \text{ with } \mathbf{x} \leqslant \mathbf{y}\} \tag{1–3}$$

$\mathcal{A}^-$ denotes the set of minimal elements in $\mathcal{A}$, and $\mathcal{U}[\mathcal{A}] = \mathcal{U}[\mathcal{A}^-]$. In terms of $\{0, 1\}^n$, a Boolean function is a mapping ψ: $\{0, 1\}^n \to \{0, 1\}$. ψ is defined by specifying either of the subsets

$$\begin{aligned} \mathcal{S}_0[\psi] &= \{\mathbf{x} \in \{0, 1\}^n\text{: } \psi(\mathbf{x}) = 0\} \\ \mathcal{S}_1[\psi] &= \{\mathbf{x} \in \{0, 1\}^n\text{: } \psi(\mathbf{x}) = 1\} \end{aligned} \tag{1–4}$$

$\mathcal{S}_0[\psi]$ and $\mathcal{S}_1[\psi]$ are called the 0-*set* and 1-*set* (0-*slice* and 1-*slice*) of ψ, respectively, and $\mathcal{S}_0[\psi]$ is the complement of $\mathcal{S}_1[\psi]$ in $\{0, 1\}^n$.

A Boolean function ψ is *increasing* if $\mathbf{x} \leqslant \mathbf{y}$ implies $\psi(\mathbf{x}) \leqslant \psi(\mathbf{y})$. If ψ is increasing, then it is a called *positive* Boolean function. ψ is positive if and only if it can be represented as a logical sum of products having no complemented variables,

$$\psi(x_1, x_2, \ldots, x_n) = \sum_i x_{i,1} x_{i,2} \cdots x_{i,n(i)} \tag{1–5}$$

A complementation-free expansion is called a *positive* expansion. If the variable set in any product of the expansion contains as a subset the set of variables in a distinct

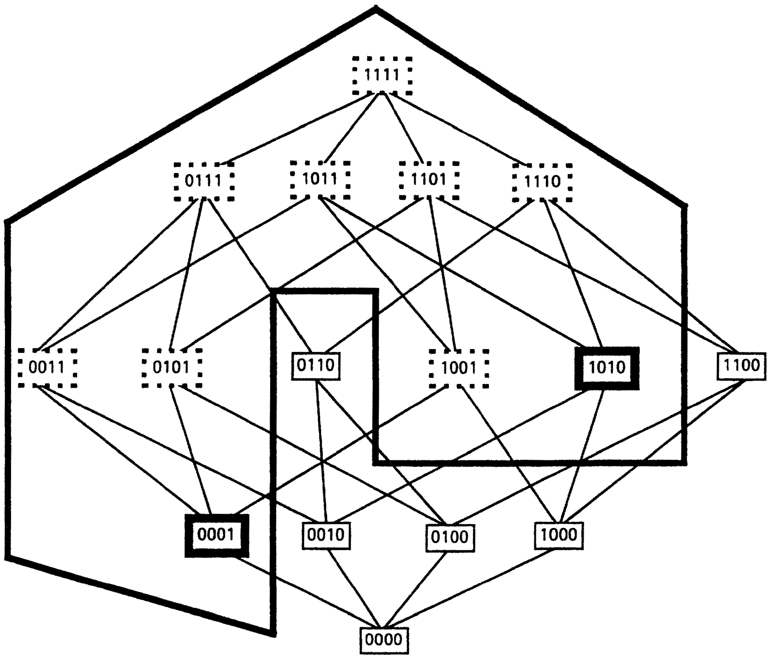


Figure 1–1. Increasing Boolean function.

product, then, whenever the former product has value 1, so too does the latter. Thus, inclusion of the former product in the expansion is redundant and it can be deleted from the expansion without changing ψ. No product whose variable set does not contain the variable set of a distinct product can be deleted without changing ψ. Performing the permitted deletions produces a unique *minimal representation* of ψ. Unless otherwise stated, it is convention to assume that positive Boolean functions are represented by positive expansions.

A Boolean function ψ is increasing if and only if there do not exist vectors $\mathbf{x} \in \mathcal{S}_1[\psi]$ and $\mathbf{y} \in \mathcal{S}_0[\psi]$ such that $\mathbf{x} \leqslant \mathbf{y}$. In terms of upper sets, ψ is increasing if and only if $\mathcal{U}[\mathcal{S}_1[\psi]^-] = \mathcal{U}[\mathcal{S}_1[\psi]] = \mathcal{S}_1[\psi]$. $\mathcal{S}_1[\psi]$ is called the *kernel* of ψ and, if ψ is increasing, then $\mathcal{S}_1[\psi]^-$ is called the *increasing basis* (or, commonly, just the *basis*) of ψ. We denote the kernel and basis by $\mathcal{K}[\psi]$ and $\mathcal{B}[\psi]$, respectively, or just $\mathcal{K}$ and $\mathcal{B}$ when not specifying the function. A four-variable increasing Boolean function is shown in Fig. 1–1. The enclosed vectors compose the kernel, and minimal elements are shown in solid black boxes. Figure 1–2 corresponds to a nonincreasing Boolean function; again the enclosed vectors compose the kernel. Since the upper set of the minimal elements does not equal the kernel, the Boolean function is nonincreasing. It becomes increasing if 0111 is switched into the kernel.

Suppose ψ is an increasing Boolean function, its positive representation according to Eq. 1–5 is assumed to be minimal, and there are m products in the expansion

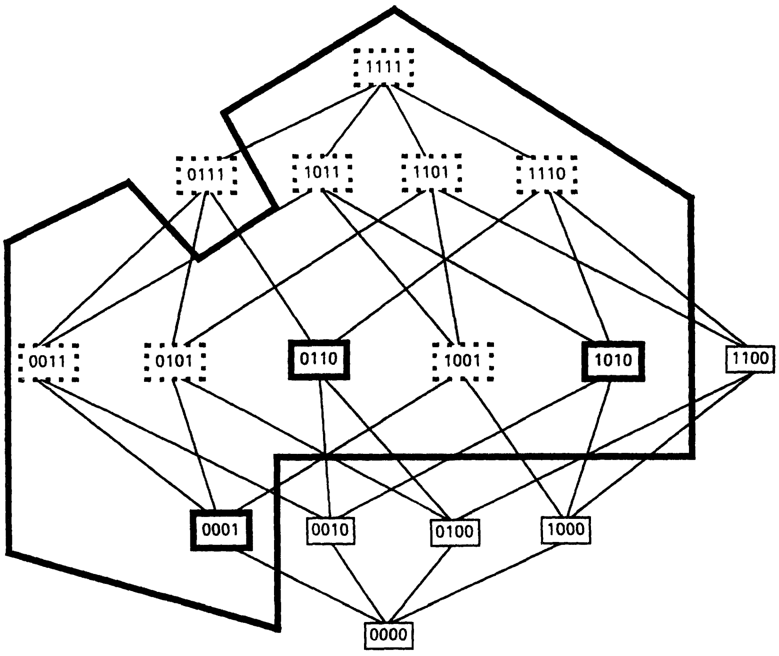


Figure 1–2. Nonincreasing Boolean function.

made up of the m variable sets

$$\begin{aligned} V_1 &= \{x_{1,1}, x_{1,2}, \ldots, x_{1,n(1)}\} \\ V_2 &= \{x_{2,1}, x_{2,2}, \ldots, x_{2,n(2)}\} \\ &\vdots \\ V_m &= \{x_{m,1}, x_{m,2}, \ldots, x_{m,n(m)}\} \end{aligned} \tag{1–6}$$

Then $\mathcal{B}[\psi] = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m\}$, where the components of $\mathbf{y}_i = (y_{i,1}, y_{i,2}, \ldots, y_{i,n})$ are

$$y_{i,k} = \begin{cases} 1, & \text{if } x_k \in V_i \\ 0, & \text{if } x_k \notin V_i \end{cases} \tag{1–7}$$

for $k = 1, 2, \ldots, n$. Hence, the product terms of the minimal expansion are referred to as the *basis elements* of ψ.

If ψ and ξ are two n-variable Boolean functions, their switching set is defined by

$$\mathcal{Z}[\psi, \xi] = \{\mathbf{x}\colon \psi(\mathbf{x}) \neq \xi(\mathbf{x})\} = (\mathcal{S}_1[\psi] \cap \mathcal{S}_0[\xi]) \cup (\mathcal{S}_0[\psi] \cap \mathcal{S}_1[\xi]) \tag{1–8}$$

If we were to switch (change the value of) $\xi(\mathbf{x})$ for every $\mathbf{x} \in \mathcal{Z}[\psi, \xi]$ or switch $\psi(\mathbf{x})$ for every $\mathbf{x} \in \mathcal{Z}[\psi, \xi]$, then we would have $\psi = \xi$. Many filtering problems are concerned with switching one Boolean function into another where there is a cost associated with switching. To formulate a switching cost, we postulate a real-valued cost function Co defined on $\{0, 1\}^n$ and define the *cost* of switching ψ into ξ or ξ into ψ by

$$Co[\psi, \xi] = \sum_{\mathbf{x} \in \mathcal{Z}[\psi, \xi]} Co(\mathbf{x}) \tag{1–9}$$

As a convention, the cost function is defined so that if the switching cost is negative, the switch is advantageous; if it is positive, the switch is disadvantageous (relative to Co). The cost of switching a function ψ into a class $\mathfrak{I}$ of functions is defined to be

$$Co_{\mathfrak{I}}[\psi] = \min\{Co[\psi, \xi]\colon\ \xi \in \mathfrak{I}\} \tag{1–10}$$

As will be discussed subsequently, for filter design an important switching cost involves constraining a filter to a particular filter class. In this case, the cost function is an error probability and the switching cost gives the increased error owing to the constraint.

1.2 MORPHOLOGICAL REPRESENTATION

Boolean functions are used to define translation-invariant windowed operators on binary digital images. These images are modeled as subsets of the Cartesian grid Z^2. Each point is called a *pixel* and is an ordered pair of integers. An image S consists of a set of pixels $z \in S$. As sets, binary images are operated upon by the usual set operations: union, intersection, complement, set subtraction, etc. Logically, an image is represented as 0s and 1s. The image value is 1 if a pixel is in the image and 0 if it is not: if $S(z)$ denotes the value of S at z, then $z \in S$ if and only if $S(z) = 1$ and $z \notin S$ if and only if $S(z) = 0$. The translation of S by pixel z is defined by $S_z = \{u + z\colon\ u \in S\}$.

To define a windowed operator, let $W = \{w_1, w_2, \ldots, w_n\}$ be an n-pixel window and ψ be an n-variable Boolean function. The corresponding set operator Ψ is defined by

$$\Psi(S)(z) = \psi(S \cap W_z) = \psi\big(S(w_1 + z), S(w_2 + z), \ldots, S(w_n + z)\big) \tag{1–11}$$

(Fig. 1–3). Note that we are simultaneously treating S and $\Psi(S)$ as subsets of the digital plane and as binary-valued functions: in the first instance, $S \cap W_z$ is the intersection between the sets S and W_z; in the second, $S \cap W_z$ is the $\{0, 1\}$-valued function S restricted to W_z. Ψ is translation-invariant, meaning $\Psi(S_z) = \Psi(S)_z$, because the same Boolean function is applied at every pixel. We call Ψ a *W-operator* and ψ its *window function*. The representation of Ψ corresponds directly to the logical representation of ψ.

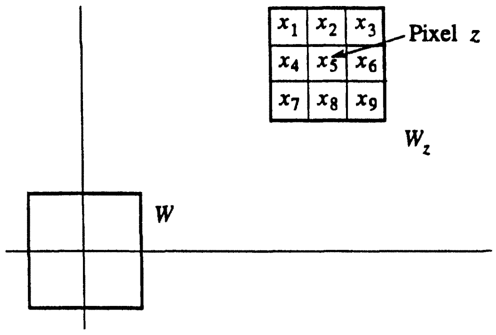


Figure 1–3. Window W applied at pixel z.

A W-operator Ψ_i defined by a single-product Boolean function

$$\psi_i(x_1, x_2, \ldots, x_n) = x_{i,1}x_{i,2} \cdots x_{i,n(i)} \tag{1–12}$$

is called an *erosion.* According to the correspondence between a binary image being defined as a subset of the grid and as a binary-valued function, $z \in \Psi_i(S)$ if and only if $\Psi_i(S)(z) = 1$. In terms of window logic, this means that the pixels in the translated window W_z corresponding to the pixels $w_{i,1}, w_{i,2}, \ldots, w_{i,n(i)}$ in W must all have value 1 so that the product of Eq. 1–12 is 1; that is, so that $x_{i,1} = x_{i,2} = \cdots = x_{i,n(i)} = 1$. The pixels in W_z corresponding to $w_{i,1}, w_{i,2}, \ldots, w_{i,n(i)}$ are $w_{i,1} + z, w_{i,2} + z, \ldots, w_{i,n(i)} + z$, and the product of Eq. 1–12 is 1 if and only if all of these pixel translates lie in S. Letting $B^i = \{w_{i,1}, w_{i,2}, \ldots, w_{i,n(i)}\}$, $z \in \Psi_i(S)$ if and only if $B^i_z \subset S$. Ψ_i is called an *erosion* operator.

Erosion of set S by set B, called a *structuring element*, is denoted by $\mathrm{E}_B(S)$ and the window function corresponding to E_B is denoted by ε_B. Figure 1–4 shows a set A and a structuring element B (with origin marked). Part (a) shows a translate of B to a pixel z for which $B_z \subset A$, so that $z \in \mathrm{E}_B(A)$; part (b) shows a translate of B to a pixel w for which $B_w \not\subset A$, so that $w \notin \mathrm{E}_B(A)$; part (c) shows $\mathrm{E}_B(A)$. Morphological image processing is based on the representation of image operators in terms of primary image operators, the most fundamental being erosion.

A set operator is said to be *increasing* if and only if $S_1 \subset S_2$ implies $\Psi(S_1) \subset \Psi(S_2)$. A W-operator Ψ with window function ψ is increasing if and only if ψ is a positive Boolean function. It follows at once from the logical representation of Eq. 1–5 that a W-operator is increasing if and only if it possesses an erosion representation [1] of the form

$$\Psi(S) = \bigcup_i \mathrm{E}_{B^i}(S) \tag{1–13}$$

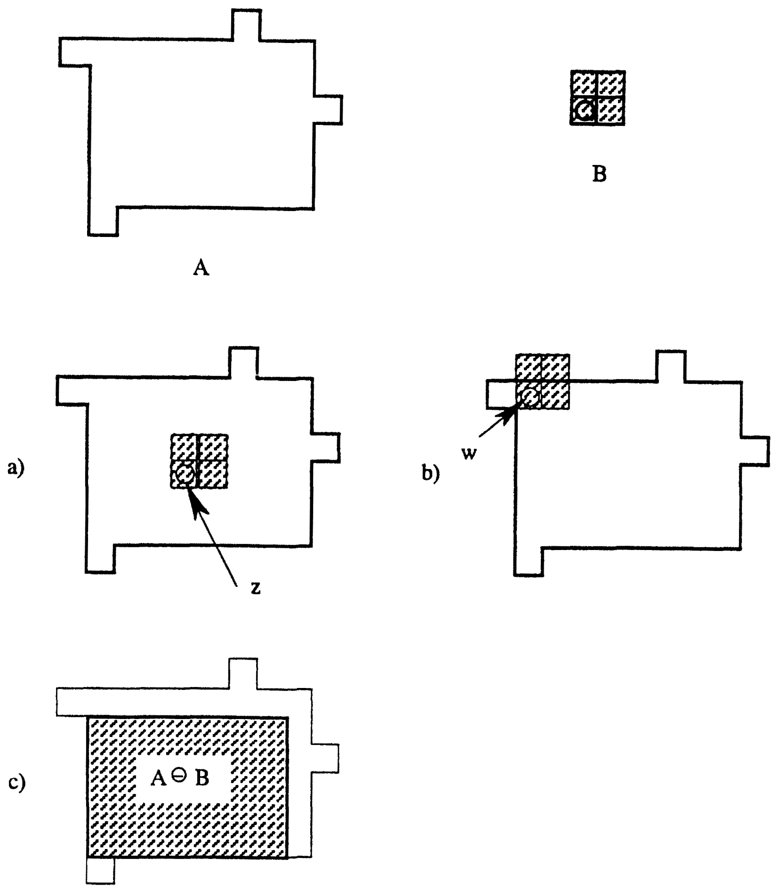


Figure 1–4. Erosion: (a) pixel in eroded set; (b) pixel not in eroded set; (c) eroded set [with $A \ominus B$ denoting erosion].

where the correspondence between Eqs. 1–5 and 1–13 is given by

$$x_{i,1}x_{i,2}\cdots x_{i,n(i)} \leftrightarrow B^i \tag{1–14}$$

The erosion representation is minimal if no structuring element is a subset of another. This corresponds directly to a minimal logical representation [2–4] and the corresponding structuring elements comprise the *basis*, $\mathcal{B}[\Psi]$, of Ψ.

Consider the binary moving median M defined over a window W containing an odd number of pixels. The window function μ is defined by $\mu(x_1, x_2, \ldots, x_n) = 1$ if and only if more than $n/2$ pixels in W are 1-valued. Hence a row of the truth table defining μ is 1-valued if and only if at least $(n+1)/2$ of the variables are 1-valued, which means that an n-variable product is a minterm in the disjunctive normal form for μ if and only if at least $(n+1)/2$ variables are uncomplemented. The minimal positive expansion for μ consists of all products of exactly $(n+1)/2$ variables. For instance, suppose W consists of the origin together with the pixels immediately below, above, to the right, and to the left of it. Let x_1, x_2, x_3, x_4, and x_5 denote the five variables corresponding to the five pixels. The minimal representation of μ is given by

$$\begin{aligned}\mu(x_1, x_2, x_3, x_4, x_5) = {} & x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_1x_2x_5 + x_1x_3x_5 \\ & + x_1x_4x_5 + x_2x_3x_4 + x_2x_3x_5 + x_2x_4x_5 + x_3x_4x_5 \qquad (1\text{–}15)\end{aligned}$$

The basis of M consists of ten structuring elements:

$$\begin{pmatrix}0&1&0\\0&\mathbf{0}&1\\0&1&0\end{pmatrix}\begin{pmatrix}0&0&0\\1&\mathbf{1}&1\\0&0&0\end{pmatrix}\begin{pmatrix}0&0&0\\1&\mathbf{1}&0\\0&1&0\end{pmatrix}\begin{pmatrix}0&0&0\\1&\mathbf{0}&1\\0&1&0\end{pmatrix}\begin{pmatrix}0&0&0\\0&\mathbf{1}&1\\0&1&0\end{pmatrix}$$
$$\begin{pmatrix}0&1&0\\1&\mathbf{1}&0\\0&0&0\end{pmatrix}\begin{pmatrix}0&1&0\\1&\mathbf{0}&1\\0&0&0\end{pmatrix}\begin{pmatrix}0&1&0\\0&\mathbf{1}&1\\0&0&0\end{pmatrix}\begin{pmatrix}0&1&0\\1&\mathbf{0}&0\\0&1&0\end{pmatrix}\begin{pmatrix}0&1&0\\0&\mathbf{1}&0\\0&1&0\end{pmatrix}$$

Note that a bold character indicates the origin and including extra 0s in the matrix representation of a structuring element does not change the structuring element.

Morphological representation of arbitrary (nonincreasing) W-operators is achieved via the *hit-or-miss transform* [5], which is defined, for any pair of disjoint subsets $E, F \subset W$, by

$$\Lambda_{E,F}(S) = \mathrm{E}_E(S) \cap \mathrm{E}_F(S^c) \qquad (1\text{–}16)$$

Pixelwise, as a binary-valued function, $\Lambda_{E,F}$ is given by

$$\begin{aligned}\Lambda_{E,F}(S)(z) &= \mathrm{E}_E(S)(z) \wedge \mathrm{E}_F(S^c)(z) \\ &= \begin{cases} 1, & \text{if } E_z \subset S \cap W_z \text{ and } F_z \subset S^c \cap W_z \\ 0, & \text{otherwise} \end{cases} \\ &= \begin{cases} 1, & \text{if } E_z \subset S \cap W_z \text{ and } F_z \cap (S \cap W_z) = \emptyset \\ 0, & \text{otherwise} \end{cases} \qquad (1\text{–}17)\end{aligned}$$

Corresponding to the logical representation of Eq. 1–2 is the *standard morphological representation* for a W-operator [6],

$$\Psi(S) = \bigcup_i \Lambda_{E^i,F^i}(S) \qquad (1\text{–}18)$$

The correspondence between Eqs. 1–2 and 1–18 is given by

$$(E^i, F^i) \leftrightarrow x_1^{p(i,1)} x_2^{p(i,2)} \cdots x_{n(i)}^{p(i,n(i))} \qquad (1\text{–}19)$$

where x_k corresponds to the pixel $w_k \in W$, x_k appears uncomplemented if $w_k \in E^i$, x_k appears complemented if $w_k \in F^i$, and x_k does not appear if $x_k \notin E^i \cup F^i$. A structuring pair is said to be *canonical* if $E^i \cup F^i = W$ and is canonical if and only if the corresponding logical product contains all n variables in the window. The representation of Eq. 1–18 is said to be canonical if all structuring pairs are canonical, in which case it corresponds to the disjunctive normal form of Eq. 1–1. If Ψ happens to be increasing, then the hit-or-miss expansion of Eq. 1–18 reduces to the erosion expansion of Eq. 1–13.

There is a useful alternative form of the hit-or-miss tranform. If L and U are binary functions defined on W such that $L \leqslant U$, then the *interval* defined by the pair (L, U) is the set of all binary functions V defined on W such that $L \leqslant V \leqslant U$. This interval is denoted by $[L, U]$ and L and U are called the *lower and upper endpoints*, respectively. For any binary image S, the *hit-or-miss transform* corresponding to (L, U) is defined at pixel z by

$$\Lambda_{L,U}(S)(z) = \begin{cases} 1, & \text{if } L_z \leqslant S \cap W_z \leqslant U_z \\ 0, & \text{otherwise} \end{cases} \tag{1–20}$$

where L_z and U_z denote L and U operating on the translate W_z. If $L = U$, then the pair is said to be canonical and $\Lambda_{L,U}(S)(z) = \Lambda_{U,U}(S)(z) = 1$ if and only if $S \cap W_z = U_z$, which means there is an exact pattern match. To see that the two definitions agree, let E and F be the sets of all pixels at which L is 1-valued and U is 0-valued, respectively; that is, $E = L$ and $F = U^c$. Then $L_z \leqslant S \cap W_z \leqslant U_z$ if and only if $E_z \subset S \cap W_z$ and $F_z \cap (S \cap W_z) = \emptyset$.

For an illustration, let W be the 3×3 square centered at the origin and

$$E = L = \begin{pmatrix} 1 & 1 & 1 \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad U = \begin{pmatrix} 1 & 1 & 1 \\ 1 & \mathbf{1} & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad F = U^c = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \mathbf{0} & 0 \\ 1 & 1 & 1 \end{pmatrix} \tag{1–21}$$

Then

$$[L, U] = \left\{ \begin{pmatrix} 1 & 1 & 1 \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 \\ 1 & \mathbf{1} & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 \\ 0 & \mathbf{1} & 1 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 \\ 1 & \mathbf{1} & 1 \\ 0 & 0 & 0 \end{pmatrix} \right\} \tag{1–22}$$

E and F are often expressed in a single matrix, with 1 and 0 denoting the pixels of E and F, respectively, and "×" denoting a (*don't care*) pixel in neither E nor F. In this notation, the hit-or-miss-transform structuring pair corresponding to (L, U) is expressed by

$$[E, F] = \begin{pmatrix} 1 & 1 & 1 \\ \times & \mathbf{1} & \times \\ 0 & 0 & 0 \end{pmatrix} \tag{1–23}$$

1.3 System Model

A key task of binary operator theory is automatic filter design. We desire an operator to optimally estimate an image when it is observed after going through some system. To frame the problem, binary digital images are modeled as discrete random sets (i.e., a collection of subsets associated with a probability distribution) [7]. If Ψ is a set operator, then for each input random set $\mathbf{S}$ there is an output random set $\Psi(\mathbf{S})$. If S is any observed realization of $\mathbf{S}$, then $\Psi(S)$ is a realization of $\Psi(\mathbf{S})$. The task is to design an operator Ψ so that, given $\mathbf{S}$, $\Psi(\mathbf{S})$ is probabilistically close to some desired process $\mathbf{I}$. We call $\mathbf{S}$, $\mathbf{I}$, and $\Psi(\mathbf{S})$ the *observation*, *ideal*, and *estimator* processes, respectively.

The distance between the ideal and estimator processes is measured by some probabilistic error measure $Er[\mathbf{I}, \Psi(\mathbf{S})]$. Assuming the operator belongs to some operator family $\Im$, an *optimal operator* relative to $\Im$ is an operator $\Psi_{opt} \in \Im$ for which

$$Er[\mathbf{I}, \Psi_{opt}(\mathbf{S})] \leqslant Er[\mathbf{I}, \Psi(\mathbf{S})] \tag{1–24}$$

for all $\Psi \in \Im$. If $\Im$ is characterized by some operator representation, then every operator $\Psi \in \Im$ has a representation and optimization can be viewed as finding the representation defining an operator possessing minimum error $Er[\mathbf{I}, \Psi(\mathbf{S})]$.

The optimization model includes a *system transformation* Ξ such that $\mathbf{S} = \Xi(\mathbf{I})$; that is, the observation process is assumed to be the output of some system operating on the ideal process [8]. Optimization involves minimizing the error $Er[\mathbf{I}, \Psi(\Xi(\mathbf{I}))]$. In general, Ξ is a multivalued image operator, meaning that, given a realization I of the ideal image, there are many possible output realizations $\Xi(I)$. The task of finding an optimal operator Ψ constitutes an inverse problem: find Ψ to best invert Ξ.

A much-studied application of the method is when Ξ is a degradation (noise) transformation — the ideal image is obscured by noise [9, 10]. The form of Ξ depends on the physical system causing the degradation. If $\mathbf{N}$ is some other binary process, then the *signal-union-noise model* is defined by the degradation transformation $\mathbf{S} = \Xi(\mathbf{I}) = \mathbf{I} \cup \mathbf{N}$. Each observed realization is of the form $S = I \cup N$, where I and N are realizations of $\mathbf{I}$ and $\mathbf{N}$, respectively. Unless there are no restrictions placed on the ideal and noise processes, the union $\mathbf{I} \cup \mathbf{N}$ does not fully define the system transformation. For instance, it might be required that $\mathbf{I}$ and $\mathbf{N}$ are statistically independent or that the union is restricted in such a way that there is no intersection between signal and noise. In document processing, the noise is often restricted to character edges, so that the noise process is strongly dependent on the signal process. Rather than union noise with the signal, one might subtract the noise, thereby having the degradation transformation $\Xi(\mathbf{I}) = \mathbf{I} - \mathbf{N}$. There could be two noise processes, $\mathbf{N}_1$ and $\mathbf{N}_2$, with the degradation both adjoining and deleting pixels and the system transformation being $\Xi(\mathbf{I}) = (\mathbf{I} \cup \mathbf{N}_1) - \mathbf{N}_2$.

Examples of morphological system models are the dilation and erosion models (see Section 3.3 for discussion of basic morphological operators). If B is a fixed structuring element and we define the system transformation by dilation or erosion, $\Xi(\mathbf{I}) = \Delta_B(\mathbf{I})$ or $\Xi(\mathbf{I}) = \mathrm{E}_B(\mathbf{I})$, then Ξ is single-valued: for each realization of the ideal process, a single determined observation results from the system. If the designed operator needs to be applied across various possible dilations or erosions, then it is better to treat the structuring element as a random set $\mathbf{B}$, in which case the dilation and erosion system transformations take the forms $\Xi(\mathbf{I}) = \Delta_\mathbf{B}(\mathbf{I})$ and $\Xi(\mathbf{I}) = \mathrm{E}_\mathbf{B}(\mathbf{I})$, respectively, both of which are multivalued. System models can be defined by other morphological operators. If $\mathbf{A}$ and $\mathbf{B}$ are random structuring elements, then the system transformation might be erosion followed by dilation, $\Xi(\mathbf{I}) = \Delta_\mathbf{A}(\mathrm{E}_\mathbf{B}(\mathbf{I}))$, or dilation followed by erosion, $\Xi(\mathbf{I}) = \mathrm{E}_\mathbf{A}(\Delta_\mathbf{B}(\mathbf{I}))$. If $\mathbf{A} = \mathbf{B}$, then these system transformations are, respectively, *opening* and *closing* by $\mathbf{B}$.

To characterize binary edge detection in terms of a system model, assume there exists a canonical edge detector: given a binary deterministic image L, there is an edge operator Θ such that $\Theta(L)$ is by definition the edge of L. For the system model, the ideal image process is a process of edges, the class of edges consisting of all possible edges that might be observed. To avoid undue mathematical complexity, assume that, for each ideal edge realization I, there exists a unique binary image R_I having edge $I[\Theta(L_I) = I]$ and we know the algorithm H that produces L_I from I. For instance, Θ might be 3×3 dilation of the image minus the image and H a contour filler. If $\Xi = \mathrm{H}$, then perfect edge construction would result from applying Θ and there would be no operator-design problem. A more practical model results by assuming the system model is H followed by noise degradation. For union noise, the system transformation takes the form $\Xi(\mathbf{I}) = \mathrm{H}(\mathbf{I}) \cup \mathbf{N}$. An optimal edge detector minimizes the error $Er[\mathbf{I}, \Psi(\mathrm{H}(\mathbf{I}) \cup \mathbf{N})]$.

For matched filtering, the ideal image is a set of points at which the object of interest is located. We assume there exists an operator that places an object to be recognized at each point of the ideal image process. An object to be recognized is a random set (shape) $\mathbf{A}$. Given a realization $I = \{z_1, z_2, \ldots, z_n\}$ of the ideal image (point set at which instances of the object are located), a realization of the observed image is given by a union of realizations $A_1, A_2, \ldots, A_n$ of $\mathbf{A}$ translated to the points $z_1, z_2, \ldots, z_n$, respectively. As a random process, the observation image is defined by the system transformation

$$\mathbf{S} = \Xi_{\mathrm{match}}(\mathbf{I}) = \bigcup_{i=1}^{N} \mathbf{A}_i + \mathbf{z}_i \tag{1–25}$$

where N is a random variable giving the number of points in the ideal image; $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N$ are random sets identically distributed to the primary shape $\mathbf{A}$;

and $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N$ are the random points composing the ideal image. The system is generally not completely characterized by these minimal conditions. For instance, there may be a constraint that objects cannot intersect or that intersection is constrained. There are also independence assumptions. Are $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N$ mutually independent? Are they independent of the points at which they are located? The randomness of $\mathbf{A}$ also needs to be modeled. It might be that $\mathbf{A}$ is a fixed shape distorted by union noise, $\mathbf{A} = A_0 \cup \mathbf{N}$; it might be that $\mathbf{A}$ is analytically described with random parameters, such as an ellipse with random axes and angle of rotation.

The system transformation of Eq. 1–25 can itself be more general. In character recognition, there exist random primary shapes (characters) different from $\mathbf{A}$, say $\mathbf{A}^1, \mathbf{A}^2, \ldots, \mathbf{A}^m$ and random point images $\mathbf{F}^1, \mathbf{F}^2, \ldots, \mathbf{F}^m$ such that

$$\Xi_{\text{match},m}(\mathbf{I}) = \left(\bigcup_{i=1}^{N} \mathbf{A}_i + \mathbf{z}_i\right) \cup \left(\bigcup_{k=1}^{m} \bigcup_{j=1}^{N(k)} \mathbf{A}_j^k + \mathbf{z}_{k,j}\right) \tag{1–26}$$

where $N(1), N(2), \ldots, N(m)$ are the numbers of points in $\mathbf{F}^1, \mathbf{F}^2, \ldots, \mathbf{F}^m$, for $k = 1, 2, \ldots, m$, $\mathbf{F}^k = \{\mathbf{z}_{k,1}, \mathbf{z}_{k,2}, \ldots, \mathbf{z}_{k,N(k)}\}$, and $\mathbf{A}_j^k$ is identically distributed to $\mathbf{A}^k$ for $j = 1, 2, \ldots, N(k)$.

Finally, suppose we desire a W-operator to emulate a given algorithm A. If A operates on a random image process $\mathbf{R}$, then the ideal image process is $\mathbf{I} = \mathrm{A}(\mathbf{R})$ and the system transformation is $\Xi_\mathrm{A}(\mathbf{I}) = \mathrm{A}^{-1}(\mathbf{I})$, where Ξ_A is multivalued because many realizations of $\mathbf{R}$ might yield the same ideal realization. The optimal emulator of A minimizes the emulation error $Er[\mathbf{I}, \Psi(\mathrm{A}^{-1}(\mathbf{I}))]$. If A is a segmentation algorithm, then $\mathbf{I}$ is composed of segmented images from $\mathbf{R}$ and $\mathrm{A}^{-1}(\mathbf{I})$ consists of unsegmented images. More generally, consider adapting a given algorithm to a noisy environment. Suppose $\mathrm{A}^{-1}(\mathbf{I})$ is the observation process resulting from algorithm inversion. As designed originally, A is meant to be applied to this process; however, suppose the observations have been degraded. Then the system transformation is given by $\Xi_\mathrm{N}(\mathbf{I}) = \mathrm{N}(\mathrm{A}^{-1}(\mathbf{I}))$, where N is a noise-degradation operator. For segmentation, if the presegmented images are degraded by erosion by a random structuring element $\mathbf{B}$, then $\Xi_\mathrm{N}(\mathbf{I}) = \mathrm{E}_\mathbf{B}(\mathrm{A}^{-1}(\mathbf{I}))$ and the optimal emulator minimizes the error $Er[\mathbf{I}, \Psi(\mathrm{E}_\mathbf{B}(\mathrm{A}^{-1}(\mathbf{I})))]$. With direct emulation using system function Ξ_A, the only advantage is translation of the algorithm into a W-operator. When emulation involves a degraded version of $\mathrm{A}^{-1}(\mathbf{I})$, there is the added advantage that the original algorithm may not work well in the degraded environment, whereas the emulation will estimate the algorithm as if it were applied in a nondegraded environment.

1.4 Optimal W-Operators

Estimation of $\mathbf{I}$ from $\Xi(\mathbf{I})$ by a W-operator Ψ requires finding a Boolean function ψ to minimize error. Since Ψ is translation-invariant, we make the modeling assumption that **I** and $\Xi(\mathbf{I})$ are jointly strict-sense stationary. This means that, if **X** is the random vector of binary values in W_z and $Y = \mathbf{I}(z)$, then the joint probability distribution for **X** and Y is independent of z, so that estimating Y from **X** yields a translation-invariant filter. We will denote random variables and random vectors by upper-case italic and bold-face letters, respectively. Realizations of the random variable Y and the random vector **X** will be denoted by y and **x**, respectively.

For operator optimization we require a loss function l: $\{0, 1\}^2 \to [0, \infty)$, where $l(a, b)$ measures the cost of the difference between a and b, with $l(0, 0) = l(1, 1) = 0$. Relative to the loss function (and owing to stationarity), filter error, $Er\langle\Psi\rangle$, is given by the expected loss from estimating $\mathbf{I}(z)$ by $\Psi(\Xi(\mathbf{I}))(z)$,

$$Er\langle\Psi\rangle = Er\big[\mathbf{I}, \Psi\big(\Xi(\mathbf{I})\big)\big] = E\big[l\big(\mathbf{I}(z), \Psi\big(\Xi(\mathbf{I})\big)(z)\big)\big] \tag{1–27}$$

where z is an arbitrary pixel. In terms of the Boolean function ψ for Ψ,

$$\begin{aligned} Er\langle\Psi\rangle &= E\big[l\big(Y, \psi(\mathbf{X})\big)\big] \\ &= \sum_{\mathbf{x}} E\big[l\big(Y, \psi(\mathbf{x})\big) \mid \mathbf{x}\big]P(\mathbf{x}) \\ &= \sum_{\{\mathbf{x}:\ \psi(\mathbf{x})=0\}} E\big[l(Y, 0) \mid \mathbf{x}\big]P(\mathbf{x}) + \sum_{\{\mathbf{x}:\ \psi(\mathbf{x})=1\}} E\big[l(Y, 1) \mid \mathbf{x}\big]P(\mathbf{x}) \\ &= \sum_{\{\mathbf{x}:\ \psi(\mathbf{x})=0\}} l(1, 0)P(Y = 1 \mid \mathbf{x})P(\mathbf{x}) \\ &\quad + \sum_{\{\mathbf{x}:\ \psi(\mathbf{x})=1\}} l(0, 1)P(Y = 0 \mid \mathbf{x})P(\mathbf{x}) \end{aligned} \tag{1–28}$$

where $P(\mathbf{x})$ denotes $P(\mathbf{X} = \mathbf{x})$. An optimal image filter is one whose Boolean function ψ minimizes Eq. 1–28. Although there can be more than one filter achieving minimal error, we shall denote "the" optimal filter and its window function by Ψ_{opt} and ψ_{opt}, respectively, the convention being that, from the standpoint of filter optimization, all filters possessing minimal error are equivalent.

The *mean-absolute-error* (*MAE*) loss function is defined by

$$l\big(y, \psi(\mathbf{x})\big) = \big|y - \psi(\mathbf{x})\big| \tag{1–29}$$

Since y and $\psi(\mathbf{x})$ are binary-valued, the loss function is given by $l(1, 0) = l(0, 1) = 1$ and $l(0, 0) = l(1, 1) = 0$. The associated error is the mean-absolute error and is denoted by $MAE\langle\Psi\rangle$. Because $l(1, 0) = l(0, 1)$, it follows from Eq. 1–28 that the optimal Boolean function and the error of the corresponding optimal set filter are given by

$$\psi_{MAE}(\mathbf{x}) = \begin{cases} 1, & \text{if } P(Y = 1 \mid \mathbf{x}) > 0.5 \\ 0, & \text{if } P(Y = 1 \mid \mathbf{x}) \leqslant 0.5 \end{cases} \tag{1–30}$$

$$\begin{aligned} MAE\langle\Psi_{MAE}\rangle &= E\big[|Y-\psi(X)|\big] \\ &= \sum_{\{\mathbf{x}:P(Y=1|\mathbf{x})>0.5\}} P(\mathbf{x})P(Y=0\mid\mathbf{x}) \\ &+ \sum_{\{\mathbf{x}:P(Y=1|\mathbf{x})\leqslant 0.5\}} P(\mathbf{x})P(Y=1\mid\mathbf{x}) \end{aligned} \tag{1–31}$$

If we list the possible realizations $\mathbf{x}$ of $\mathbf{X}$ in a table along with the prior probabilities $P(\mathbf{x})$ and conditional probabilities $P(Y=0\mid\mathbf{x})$ and $P(Y=1\mid\mathbf{x})$, then $MAE\langle\Psi_{MAE}\rangle$ is obtained by summing the joint probabilities corresponding to the values (0 or 1) not chosen for ψ_{MAE}. Because images are binary,

$$E[Y\mid\mathbf{X}]=P(Y=1\mid\mathbf{X}) \tag{1–32}$$

so that

$$\psi_{MAE}(\mathbf{x})=\begin{cases}1, & \text{if } E[Y\mid\mathbf{x}]>0.5\\ 0, & \text{if } E[Y\mid\mathbf{x}]\leqslant 0.5\end{cases} \tag{1–33}$$

the *binary conditional expectation*. Relative to the input image $\mathbf{S}=\Xi(\mathbf{I})$, the optimal MAE filter is defined pixelwise by

$$\Psi_{MAE}(\mathbf{S})(z)=\begin{cases}1, & \text{if } P(\mathbf{I}(z)=1\mid\mathbf{S}\cap W_z)>0.5\\ 0, & \text{if } P(\mathbf{I}(z)=1\mid\mathbf{S}\cap W_z)\leqslant 0.5\end{cases} \tag{1–34}$$

The MAE loss function is used in many applications. Consider object recognition in a model in which $\mathbf{I}$ gives the locations of a shape A. The conditional probability in Eq. 1–34 gives the probability of A being at pixel z given the observation $\mathbf{S}\cap W_z$. Equivalently, the shape occurrence probability is given by $P(Y=1\mid\mathbf{X})$. Now, suppose there exists a single observation $\mathbf{x}_A$ for which $P(Y=1\mid\mathbf{x}_A)>0.5$ and $P(Y=1\mid\mathbf{x})<0.5$ for $\mathbf{x}\neq\mathbf{x}_A$. Then, according to Eq. 1–30, the optimal MAE filter is defined by the Boolean function

$$\psi_{MAE}(\mathbf{x})=\begin{cases}1, & \text{if } \mathbf{x}=\mathbf{x}_A\\ 0, & \text{if } \mathbf{x}\neq\mathbf{x}_A\end{cases} \tag{1–35}$$

In particular, suppose $P(Y=1\mid\mathbf{x}_A)=1$ and $P(Y=1\mid\mathbf{x})<0.5$ for $\mathbf{x}\neq\mathbf{x}_A$. Then $\psi_{MAE}(\mathbf{x})=1$ if, based on the observation $\mathbf{x}$, we are almost sure that shape A occurs. If, however, there were to exist $\mathbf{x}_0\neq\mathbf{x}_A$ such that $P(Y=1\mid\mathbf{x}_0)>0.5$, then we would also have $\psi_{MAE}(\mathbf{x}_0)=1$.

A different loss function can achieve a more general shape recognition result. Let $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_m$ denote the $m=2^n$ possible realizations of $\mathbf{X}$. Let $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_r$, $r<m$, correspond to the patterns which when observed guarantee (almost surely)

that the shape A occurs at z and $\mathbf{x}_{r+1}, \mathbf{x}_{r+2}, \ldots, \mathbf{x}_m$ correspond to patterns which when observed do not guarantee (almost surely) that A occurs at z. In terms of conditional probabilities, $P(Y = 1 \mid \mathbf{x}_j) = 1$ for $j = 1, 2, \ldots, r$ and $P(Y = 1 \mid \mathbf{x}_j) < 1$ for $j = r + 1, \ldots, m$. Let

$$\tau = \max_{j>r} P(Y = 1 \mid \mathbf{x}_j)P(\mathbf{x}_j) < 1 \tag{1–36}$$

and define a loss function by $l(0, 0) = l(1, 1) = 0$, $l(1, 0) = 1$, and $l(0, 1) = (1 - \tau)^{-1}$. We claim that the optimal shape-recognition filter relative to l is given by

$$\psi_{SR}(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} = \mathbf{x}_j,\ j = 1,\ 2, \ldots, r \\ 0, & \text{if } \mathbf{x} = \mathbf{x}_j,\ j = r + 1, \ldots, m \end{cases} \tag{1–37}$$

which is precisely the filter we desire. From Eq. 1–28, the error for ψ_{SR} is given by

$$\varepsilon_{SR} = \sum_{j=r+1}^{m} P(Y = 1 \mid \mathbf{x}_j)P(\mathbf{x}_j) \tag{1–38}$$

Now, suppose ψ_{SR} is changed so that some of the first r vectors, say $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_b$, are redefined to be 0-valued and some of the originally 0-valued vectors, say $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c$, are redefined to be 1-valued. The error for the new function, call it ξ, is given by

$$\begin{aligned} \varepsilon_\xi &= \varepsilon_{SR} - \sum_{j=1}^{c} P(Y = 1 \mid \mathbf{v}_j)P(\mathbf{v}_j) \\ &\quad + \frac{1}{1 - \tau} \sum_{j=1}^{c} P(Y = 0 \mid \mathbf{v}_j)P(\mathbf{v}_j) + \sum_{j=1}^{b} P(\mathbf{u}_j) \end{aligned} \tag{1–39}$$

The first sum is bounded above by $c\tau$, the second sum is bounded below by $c(1 - \tau)$, and the third sum is nonnegative. Hence,

$$\varepsilon_\xi \geqslant \varepsilon_{SR} + c(1 - \tau) + \sum_{j=1}^{b} P(\mathbf{u}_j) > \varepsilon_{SR} \tag{1–40}$$

thereby proving that ψ_{SR} is optimal relative to the new loss function.

Every W-operator Ψ can be canonically represented as

$$\Psi(S) = \bigcup_{(E,F)\in\mathcal{C}_\Psi} \Lambda_{E,F}(S) \tag{1–41}$$

where $\mathcal{C}_\Psi$ is the set of canonical structuring pairs defining Ψ. Owing to canonicalness, $\Psi(S)(z) = 1$ if and only if there is a structuring pair $(E, F) \in \mathcal{C}_\Psi$ such that $E = S \cap W_z$ and $F = S^c \cap W_z$. We denote the structuring-pair set defining an optimal filter Ψ_{opt} by $\mathcal{C}_{opt}$.

For an arbitrary loss function [8] $(E, F) \in \mathcal{C}_{opt}$ if and only if

$$E\big[l(\mathbf{I}(z), 1) \mid \mathbf{S} \cap W_z = E\big] < E\big[l(\mathbf{I}(z), 0) \mid \mathbf{S} \cap W_z = E\big] \tag{1–42}$$

where $\mathbf{S} \cap W_z = E$ means that the observed windowed process equals E and, because the structuring pair is canonical, $\mathbf{S}^c \cap W_z = F$. Each canonical pair corresponds to the binary n-vector $\mathbf{x}$ of values in (E, F); for each set operator Ψ with window function ψ, $\mathcal{C}_\Psi$ corresponds to the kernel $\mathcal{K}[\psi]$; and Eq. 1–42 is equivalent to $\mathbf{x} \in \mathcal{K}[\psi_{opt}]$ if and only if

$$E\big[l(Y, 1) \mid \mathbf{x}\big] < E\big[l(Y, 0) \mid \mathbf{x}\big] \tag{1–43}$$

Once the disjunctive normal form for ψ_{opt} (canonical expansion for Ψ_{opt}) has been determined, logic reduction can be employed to reduce the expansion to a non-canonical sum of products (hit-or-miss expansion) involving fewer logic gates. For estimation purposes, we continue to assume canonical representation.

For the MAE loss function [11] $\mathcal{C}_{opt}$ is denoted by $\mathcal{C}_{MAE}$ and

$$E\big[l(\mathbf{I}(z), 1) \mid \mathbf{S} \cap W_z = E\big] = 1 - P\big(\mathbf{I}(z) = 1 \mid \mathbf{S} \cap W_z = E\big) \tag{1–44}$$

According to Eq. 1–42, $(E, F) \in \mathcal{C}_{MAE}$ if and only if

$$P\big(\mathbf{I}(z) = 1 \mid \mathbf{S} \cap W_z = E\big) > 0.5 \tag{1–45}$$

Equivalently, $\mathbf{x} \in \mathcal{K}[\psi_{MAE}]$ if and only if

$$P(Y = 1 \mid \mathbf{x}) > 0.5 \tag{1–46}$$

We often use a suboptimal filter Ψ instead of the optimal filter Ψ_{opt}, with a concomitant increase in error. The expansion of Eq. 1–41 is taken over $\mathcal{C}_\Psi$ instead of $\mathcal{C}_{opt}$ and, equivalently, $\mathcal{K}[\psi]$ replaces $\mathcal{K}[\psi_{opt}]$. To quantify the error increase, if $P(\mathbf{x}) > 0$, we define the *advantage* of $\mathbf{x}$ by

$$Ad_l(\mathbf{x}) = \big(E\big[l(Y, 0) \mid \mathbf{x}\big] - E\big[l(Y, 1) \mid \mathbf{x}\big]\big)P(\mathbf{x}) \tag{1–47}$$

According to Eq. 1–43, $Ad_l(\mathbf{x}) > 0$ if and only if $\mathbf{x} \in \mathcal{K}[\psi_{opt}]$. An increase in error can arise in two ways from using Ψ instead of Ψ_{opt}: $\mathbf{x} \in \mathcal{K}[\psi_{opt}]$ but $\mathbf{x} \notin \mathcal{K}[\psi]$, or $\mathbf{x} \notin \mathcal{K}[\psi_{opt}]$ but $\mathbf{x} \in \mathcal{K}[\psi]$. This error increase is the switching cost with cost

Table 1.1. MAE cost of using the median instead of the optimal filter.

$\mathbf{x}$	$P(\mathbf{x})$	$P(Y=1 \mid \mathbf{x})$	$P(Y=0 \mid \mathbf{x})$	$Ad_{MAE}(\mathbf{x})$
000	0.30	0.10	0.90	−0.24
001	0.05	0.10	0.90	−0.04
010	0.20	0.60	0.40	0.04
011	0.05	0.70	0.30	0.02
100	0.05	0.10	0.90	−0.04
101	0.05	0.30	0.70	−0.02
110	0.10	0.70	0.30	0.04
111	0.20	0.90	0.10	0.16

function $Co(\mathbf{x}) = |Ad_l(\mathbf{x})|$. According to Eq. 1–9, the total error increase from using Ψ instead of Ψ_{opt} is

$$\begin{aligned} Er\langle\Psi\rangle - Er\langle\Psi_{opt}\rangle &= Co[\psi, \psi_{opt}] \\ &= \sum_{\mathbf{x}\in\mathcal{Z}[\psi,\psi_{opt}]} |Ad_l(\mathbf{x})| = \sum_{\mathbf{x}\in\mathcal{K}[\psi_{opt}]\Delta\mathcal{K}[\psi]} |Ad_l(\mathbf{x})| \end{aligned} \tag{1–48}$$

where the last sum is over the symmetric difference between the kernels. $Ad_l(\mathbf{x})$ can also be written as $Ad_l(E, F)$, where the canonical pair (E, F) corresponds to $\mathbf{x}$.

Using a particular filter without concern for optimization almost always leads to suboptimality and often the error increase is excessive. To illustrate the cost of *ad hoc* filter selection, we consider a three-point horizontal window centered at the origin and the MAE loss function. There are eight observation vectors. For the probabilities and advantages in Table 1.1, $\mathcal{K}[\psi_{opt}] = \{010, 011, 110, 111\}$. Suppose, instead of using Ψ_{opt}, one were to apply the three-point median M, whose window function has kernel $\mathcal{K}[\mu] = \{011, 101, 110, 111\}$. Since $\mathcal{K}[\psi_{opt}]\Delta\mathcal{K}[\mu] = \{010, 101\}$, Eq. 1–48 gives the MAE cost of using the median instead of the optimal filter as 0.06.

Suboptimality can be introduced when there is too much logic for implementation, even after reduction. All pairs in C_{opt} contribute to error reduction but some contribute very little and can be deleted with a small loss in filter performance. If $C_\Psi \subset C_{opt}$, then

$$Er\langle\Psi\rangle - Er\langle\Psi_{opt}\rangle = \sum_{\mathbf{x}\in\mathcal{K}[\psi_{opt}]-\mathcal{K}[\psi]} |Ad_l(\mathbf{x})| \tag{1–49}$$

In practice, vectors with positive advantage are listed from largest to smallest advantage. Filter logic is reduced by deleting those at the bottom of the list prior to logic reduction.

For the MAE loss function, advantage and absolute advantage are given by

$$Ad_{MAE}(\mathbf{x}) = \big[P(Y=1 \mid \mathbf{x}) - P(Y=0 \mid \mathbf{x})\big]P(\mathbf{x}) \tag{1–50}$$

$$\big|Ad_{MAE}(\mathbf{x})\big| = \big|1 - 2P(Y=1 \mid \mathbf{x})P(\mathbf{x})\big| \tag{1–51}$$

In image restoration, $Ad_{MAE}(\mathbf{x})$ $[Ad_{MAE}(E, F)]$ has been called [9, 10] the *restoration effect* of $\mathbf{x}$ $[(E, F)]$. The advantage for the SR loss function is

$$Ad_{SR}(\mathbf{x}) = \big[P(Y=1 \mid \mathbf{x}) - (1-\tau)^{-1}P(Y=0 \mid \mathbf{x})\big]P(\mathbf{x}) \tag{1–52}$$

If $P(Y = 1 \mid \mathbf{x}) = 1$, then $Ad_{SR}(\mathbf{x}) = 1$; if $P(Y = 1 \mid \mathbf{x}) < 1$, then $Ad_{SR}(\mathbf{x}) \leqslant \tau - 1 < 0$.

1.5 Estimation of Optimal *W*-Operators

In practice, the optimal filter is statistically estimated from image realizations by estimating the conditional expectations composing the decision criterion of Eq. 1–43. This is accomplished by taking image-pair realizations (I_1, S_1), $(I_2, S_2), \ldots, (I_m, S_m)$ of $\mathbf{I}$ and $\mathbf{S} = \Xi(\mathbf{I})$ and forming estimators

$$\begin{aligned} \hat{e}_{l,\mathbf{x}}(0) &= \widehat{E}\big[l(Y,0) \mid \mathbf{x}\big] \\ \hat{e}_{l,\mathbf{x}}(1) &= \widehat{E}\big[l(Y,1) \mid \mathbf{x}\big] \end{aligned} \tag{1–53}$$

The designed estimate of the optimal filter, $\widehat{\Psi}_{opt}$ (with window function $\widehat{\psi}_{opt}$), is determined by the set $\mathcal{K}[\widehat{\psi}_{opt}]$ of observation vectors $\mathbf{x}$ for which

$$\hat{e}_{l,\mathbf{x}}(1) < \hat{e}_{l,\mathbf{x}}(0) \tag{1–54}$$

For a given $\mathbf{x}$ there is an increase in error owing to estimation of the optimal filter if and only if the inequality of Eq. 1–43 holds but the inequality of Eq. 1–54 does not, or if Eq. 1–43 does not hold but Eq. 1–54 does. These cases correspond to two types of estimation error: $\mathbf{x} \in \mathcal{K}[\psi_{opt}]$ but $\mathbf{x} \notin \mathcal{K}[\widehat{\psi}_{opt}]$ and $\mathbf{x} \notin \mathcal{K}[\psi_{opt}]$ but $\mathbf{x} \in \mathcal{K}[\widehat{\psi}_{opt}]$. This suboptimality situation is covered by Eq. 1–48 with $\widehat{\Psi}_{opt}$ being the suboptimal filter used in place of Ψ. Hence, $Er\langle\widehat{\Psi}_{opt}\rangle - Er\langle\Psi_{opt}\rangle$, the error increase owing to estimation of the optimal filter is given by Eq. 1–48 with $\mathcal{K}[\widehat{\psi}_{opt}]$ in place of $\mathcal{K}[\psi]$.

For the MAE loss function, we can employ Eq. 1–42 or the respective equivalent conditions of Eqs. 1–44 and 1–45. For the latter, we can use the probability estimator

$$\widehat{P}(Y = k \mid \mathbf{x}) = \frac{Card[Y = k \mid \mathbf{x}]}{Card[\mathbf{x}]} \tag{1–55}$$

for $k = 0, 1$, where *Card* denotes set cardinality and the numerator and denominator give the number of times the sample ideal images are k-valued given $\mathbf{x}$ and the number of times $\mathbf{x}$ is observed across the sample, respectively. $\mathbf{x} \in \mathcal{K}[\widehat{\psi}_{MAE}]$ if and only if Eq. 1–46 holds with the estimate of Eq. 1–55 used in place of the true conditional probability.

As thus far stated, the error increase corresponds to a given designed kernel $\mathcal{K}[\widehat{\psi}_{opt}]$; in fact, $\mathcal{K}[\widehat{\psi}_{opt}]$ is a random collection depending on the realizations selected. Thus, $Er\langle\widehat{\Psi}_{opt}\rangle - Er\langle\Psi_{opt}\rangle$ is a random variable and we measure the precision with which $\widehat{\Psi}_{opt}$ estimates Ψ_{opt} by the expected error increase,

$$\rho_{opt} = E\left[Er\langle\widehat{\Psi}_{opt}\rangle - Er\langle\Psi_{opt}\rangle\right] \tag{1–56}$$

Note that $\mathbf{x} \in \mathcal{K}[\psi_{opt}] - \mathcal{K}[\widehat{\psi}_{opt}]$ if and only if $Ad_l(\mathbf{x}) > 0$ and $\hat{e}_{l,\mathbf{x}}(1) \geqslant \hat{e}_{l,\mathbf{x}}(0)$. Moreover, $\mathbf{x} \in \mathcal{K}[\widehat{\psi}_{opt}] - \mathcal{K}[\psi_{opt}]$ if and only if $Ad_l(\mathbf{x}) \leqslant 0$ and $\hat{e}_{l,\mathbf{x}}(1) < \hat{e}_{l,\mathbf{x}}(0)$. Hence,

$$\begin{aligned}\rho_{opt} = {} & \sum_{\mathbf{x}\in\mathcal{K}[\psi_{opt}]} Ad_l(\mathbf{x})P\left(\hat{e}_{l,\mathbf{x}}(1) \geqslant \hat{e}_{l,\mathbf{x}}(0)\right) \\ & - \sum_{\mathbf{x}\notin\mathcal{K}[\psi_{opt}]} Ad_l(\mathbf{x})P\left(\hat{e}_{l,\mathbf{x}}(1) < \hat{e}_{l,\mathbf{x}}(0)\right)\end{aligned} \tag{1–57}$$

This error has been studied in the context of binary-image restoration and the analysis applies to the general MAE theory [9, 10, 12]. For the MAE loss function,

$$\begin{aligned}\rho_{MAE} = {} & \sum_{\{\mathbf{x}\in P(Y=1|\mathbf{x})>0.5\}} |2P(Y=1\mid\mathbf{x})-1|P\left(\widehat{P}(Y=1\mid\mathbf{x})\leqslant 0.5\right)P(\mathbf{x}) \\ & + \sum_{\{\mathbf{x}\in P(Y=1|\mathbf{x})\leqslant 0.5\}} |2P(Y=1\mid\mathbf{x})-1|P\left(\widehat{P}(Y=1\mid\mathbf{x}) > 0.5\right)P(\mathbf{x})\end{aligned} \tag{1–58}$$

As the number of observations increases to infinity, $\rho_{MAE} \to 0$.

The preceding estimation methodology is somewhat idealized. If $P(\mathbf{x}) > 0$, then we obtain good estimates $\hat{e}_{l,\mathbf{x}}(0)$ and $\hat{e}_{l,\mathbf{x}}(1)$ for sufficiently large observation samples and the error analysis applies. In practice, however, a particular structuring pair $\mathbf{x}$ may not be observed during the estimation procedure, in which case $\hat{e}_{l,\mathbf{x}}(0)$ and $\hat{e}_{l,\mathbf{x}}(1)$ are not defined. Then, based solely on our statistical knowledge, it is irrelevant whether $\mathbf{x}$ is placed into $\mathcal{K}[\widehat{\psi}_{opt}]$. If $\hat{e}_{l,\mathbf{x}}(0)$ and $\hat{e}_{l,\mathbf{x}}(1)$ are not defined, then the design procedure discussed in the next section decides whether to place $\mathbf{x}$ into $\mathcal{K}[\widehat{\psi}_{opt}]$ on the basis of algorithm efficiency. In fact, if $P(\mathbf{x}) > 0$, then whether or not $\mathbf{x} \in \mathcal{K}[\widehat{\psi}_{opt}]$ does affect the error of estimation, the arbitrariness of our decision coming from lack of statistical knowledge (too small a sample). Extra criteria can be employed to decide whether an unobserved structuring pair is placed into $\mathcal{K}[\widehat{\psi}_{opt}]$. Such criteria produce a different design procedure and therefore different errors of estimation. A key design method for document processing is the use of differencing representation, which produces the estimate $\widehat{\Psi}_{opt}$ from a different representation than the standard hit-or-miss expansion [9]. For some system models, error of estimation is less for differencing design; for others, it is less for standard hit-or-miss design. Even if a structuring pair is observed, it may be observed so few times (a single observation not being unusual) that we lack confidence in the estimates $\hat{e}_{l,\mathbf{x}}(0)$ and $\hat{e}_{l,\mathbf{x}}(1)$. In such a case we might employ a conditonal decision criterion of the following form: if $\mathbf{x}$ is observed at least r times, then decide whether to place $\mathbf{x}$ into $\mathcal{K}[\widehat{\psi}_{opt}]$ on the basis of Eq. 1–54; otherwise, apply some other stated criterion. Such conditional decision criteria have their own errors of estimation [9].

1.6 Design Procedure

The design procedure of set operators for binary image analysis is composed of two main steps: (1) estimation of conditional probabilities and (2) computation of an operator of minimal cost in accordance with a given loss function and the estimated conditional probabilities. The detailed design procedure may be outlined as follows:

1. Shift the window to all pixel locations within the observation image.
2. At each location, record the observed canonical structuring pair.
3. At each location, record the value of the pixel in the ideal image that is colocated with the window origin in the observation image.
4. For each structuring pair, tally the number of times a 1 is observed in the ideal image and the number of times a 0 is observed.
5. For the operator representation select, for each canonical structuring pair observed, the value (0 or 1) of minimal cost.
6. To reduce the representation cost, perform logic minimization assuming that the unobserved templates are "don't cares".

Except for the fact that a large sample may be required, the computational cost of the first five steps is low; however, the computational cost of the sixth step may be high.

Step 5 creates a truth table defining a family of statistically equivalent Boolean functions. Each function in the family possesses a large number of representations. An ideal procedure for logical reduction would give the best representation (i.e., the one using a minimal number of logic gates) among all possible representations of all equivalent functions. In practical applications, such a procedure is usually not available. Typically, the canonical representation of one (or, at best, several) of the equivalent Boolean functions is chosen and the best representation for this function is found.

In this more modest context, simplification of a Boolean function consists of transforming a given expression into another expression with fewer terms (products) or literals (variables). A Boolean expression is considered a *minimal expression* if: (1) there does not exist an equivalent Boolean expression with a smaller number of terms or (2) there does not exist an equivalent expression with an equal number of terms but with a smaller number of literals. Boolean expressions can be simplified or minimized via the laws of Boolean algebra. We consider two reduction algorithms. Intervals in $\{0, 1\}^n$ play a key role. If $\mathbf{a}, \mathbf{b} \in \{0, 1\}^n$ and $\mathbf{a} \leqslant \mathbf{b}$, then the *interval* (*cube*) $[\mathbf{a}, \mathbf{b}] = \{\mathbf{x} \in \{0, 1\}^n\colon\ \mathbf{a} \leqslant \mathbf{x} \leqslant \mathbf{b}\}$. The dimension of $[\mathbf{a}, \mathbf{b}]$, $dim([\mathbf{a}, \mathbf{b}])$, is the difference between the number of 1-valued components in $\mathbf{b}$ and $\mathbf{a}$. A cube of dimension n is called an n-cube.

A classical simplification algorithm for Boolean expressions is the *Quine-McCluskey* (*QM*) tabular algorithm [13, 14]. This algorithm provides a minimal expression and is well suited for computer implementation since it systematizes the simplification process. The QM algorithm is composed of three basic steps:

1. Complete with 1s the table describing the Boolean function.
2. Take the set of 1-valued patterns as vertices (points) in an n-dimensional space and perform a systematic minimization: from the set of vertices (0-cubes), try to find all adjacent vertices (1-cubes); from the set of 1-cubes, try to find all adjacent 1-cubes (2-cubes); and so on. The set of cubes resulting at the final step of the process is called the set of *prime implicants*.
3. Select the essential prime implicants from the set of prime implicants generated in step 2. A term is not essential in the set of prime implicants if it can be represented by two other terms of the set. The central point in this step is to compute the smallest subset of the set of prime implicants that is sufficient to represent the function.

Figure 1–5 illustrates the generation of the prime-implicant set in a simple example. The drawback of the QM algorithm is that step 1 generates an enormous amount of data, even when the number of known points in the function is small relative to the size of the domain. The QM algorithm cannot be practically used for functions of more than twelve variables.

Incremental splitting of intervals (*ISI algorithm*) has been proposed for functions having large numbers of variables and relatively small numbers of fixed values, a common occurrence when learning set operators [8, 15]. The main idea of the

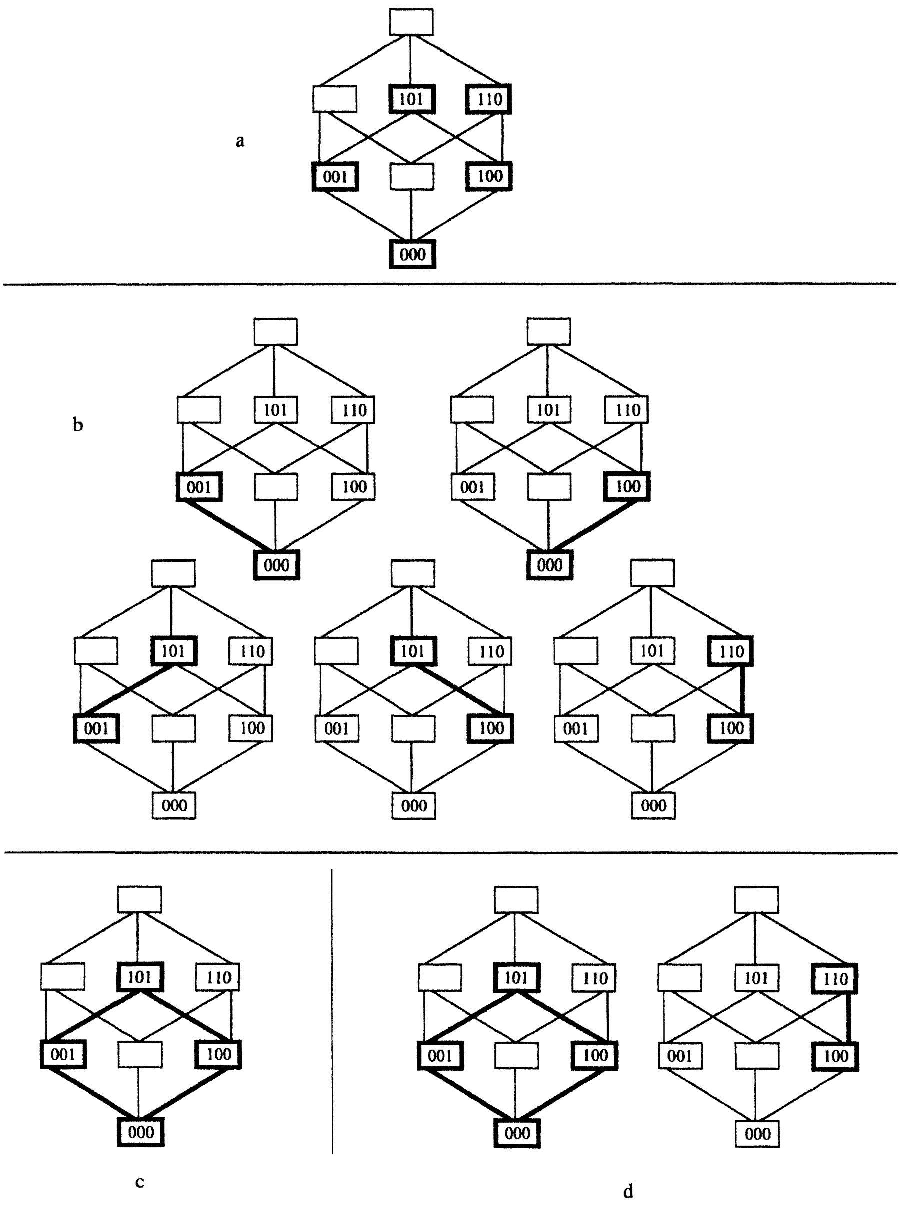


Figure 1–5. Generation of prime-implicant set by QM algorithm.

QM algorithm is repetition of a process that joins cubes (from the 0-cubes) to obtain cubes as large as possible. The main idea of the ISI algorithm is dual to this: it is a repeated process that splits cubes into lower order cubes while there are negative examples that must be satisfied. Successive application of this procedure, with some basic caveats, results in a minimal expression.

The ISI algorithm is composed of five basic steps:

1. Let $\mathcal{T}$ be an initial set of prime implicants. Usually, in a space of dimension n, $\mathcal{T}$ is the set whose single element is the n-cube (the set $\{0, 1\}^n$). If there are no negative examples, then the algorithm stops and $\mathcal{T}$ itself is the output.
2. Separate the cubes in $\mathcal{T}$ according to the following criterion: put the cubes of $\mathcal{T}$ that cover the next negative example (shape associated to 0) to be extracted in $\mathcal{N}$ and the others in $\mathcal{P}$. Set $\mathcal{M} = \emptyset$.
3. Split each cube in $\mathcal{N}$ and represent it by its low-order cubes. The low-order cubes that were not covered by some cube in $\mathcal{P}$ are put in $\mathcal{M}$.
4. Put $\mathcal{T} = \mathcal{P} \cup \mathcal{M}$ and repeat steps 2 and 3 for the next negative example, while negative examples exist.
5. Select the essential prime implicants by the method used in the QM algorithm.

Symbolically, the first four steps of the algorithm can be expressed in the following manner, where all vectors have n binary components, $\mathbf{0}$ is the zero vector, $\mathbf{1}$ is the vector of all 1s, and $[\mathbf{a}, \mathbf{b}]$ is the cube in $\{0, 1\}^n$ determined by $\mathbf{a}$ and $\mathbf{b}$:

```
1. Set $\mathcal{T} = \{[\mathbf{0}, \mathbf{1}]\}$;
2. Set $\mathcal{X} = \{\mathbf{x} \in \{0, 1\}^n$: $(\mathbf{x}, 0)$ is a line of the truth table$\}$;
3. If $\mathcal{X} \neq \emptyset$, select $\mathbf{x} \in \mathcal{X}$, else go to 11;
4. Set $\mathcal{N} = \{[\mathbf{a}, \mathbf{b}] \in \mathcal{T}\colon \mathbf{x} \in [\mathbf{a}, \mathbf{b}]\}$;
5. Set $\mathcal{P} = \mathcal{T} - \mathcal{N}$;
6. Set $\mathcal{M} = \emptyset$;
7. For each $[\mathbf{a}, \mathbf{b}] \in \mathcal{N}$:
    • Set $\mathcal{S}$ = split of $[\mathbf{a}, \mathbf{b}]$ by $\mathbf{x}$;
    • Set $\mathcal{M} = \mathcal{M} \cup \{[\mathbf{c}, \mathbf{d}] \in \mathcal{S}\colon \exists$ no $[\mathbf{u}, \mathbf{v}] \in \mathcal{P}$ such that $[\mathbf{c}, \mathbf{d}] \subset [\mathbf{u}, \mathbf{v}]\}$;
8. Set $\mathcal{T} = \mathcal{P} \cup \mathcal{M}$;
9. Set $\mathcal{X} = \mathcal{X} - \{\mathbf{x}\}$;
10. Go to 3;
11. Return $\mathcal{T}$;
12. END.
```

A fundamental aspect of the ISI algorithm is splitting a cube by a negative example. A cube $[\mathbf{a}, \mathbf{b}]$ is *split* by a negative example $\mathbf{x} \in [\mathbf{a}, \mathbf{b}]$ into a collection of intervals $\mathcal{S}$ in the following manner:

$$\mathcal{S} = \left\{\left[\mathbf{a}, \mathbf{b} \wedge \mathbf{z}_1'\right]\colon \mathbf{z}_1 \leqslant \mathbf{x}\right\} \cup \left\{[\mathbf{a} \vee \mathbf{z}_1, \mathbf{b}]\colon \mathbf{z}_1 \leqslant \mathbf{x}'\right\} \tag{1–59}$$

where $\mathbf{z}_1$ denotes a generic vector having exactly one 1-valued component. It can be proved that

$$Card(S) = dim([\mathbf{a}, \mathbf{b}]) \tag{1-60}$$

where the number on the left is a count of intervals and the number on the right is a count of elements. Moreover, for any interval $[\mathbf{u}, \mathbf{v}] \in S$,

$$dim([\mathbf{u}, \mathbf{v}]) = dim([\mathbf{a}, \mathbf{b}]) - 1 \tag{1-61}$$

Thus, the split of an n-cube by a negative example produces n cubes of dimension $n - 1$. It can be also proved that the intervals of S are maximal.

The ISI algorithm is recursive. Each execution of step 8 gives a set of prime implicants. Iterations of the algorithm can be visualized via a tree, where nodes at a given level are cubes resulting after the execution of the iteration. If we consider the root at level zero, then the nodes in the first level are the cubes resulting after the extraction of the first negative example and so on. The dynamics for minimization of a function of three variables are shown in Fig. 1–6. The same 3-space process can be viewed in Fig. 1–7.

For small numbers of variables the ISI algorithm uses much less storage space than the QM algorithm; however, relative execution time depends on the quantity and distribution of the examples given. The ISI algorithm can be applied for minimizing expressions with large numbers of variables that cannot be treated by the QM algorithm so long as the number of don't cares is also large [8].

We now present some example applications [8] of optimal design of set operators in binary image analysis using the MAE loss function and the ISI learning algorithm. We begin with an example that illustrates the complete design process.

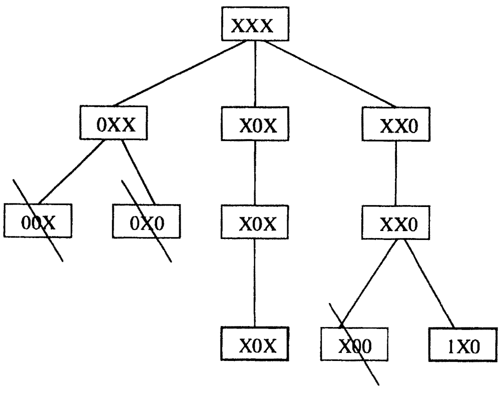


Figure 1–6. Minimization of a function of three variables.

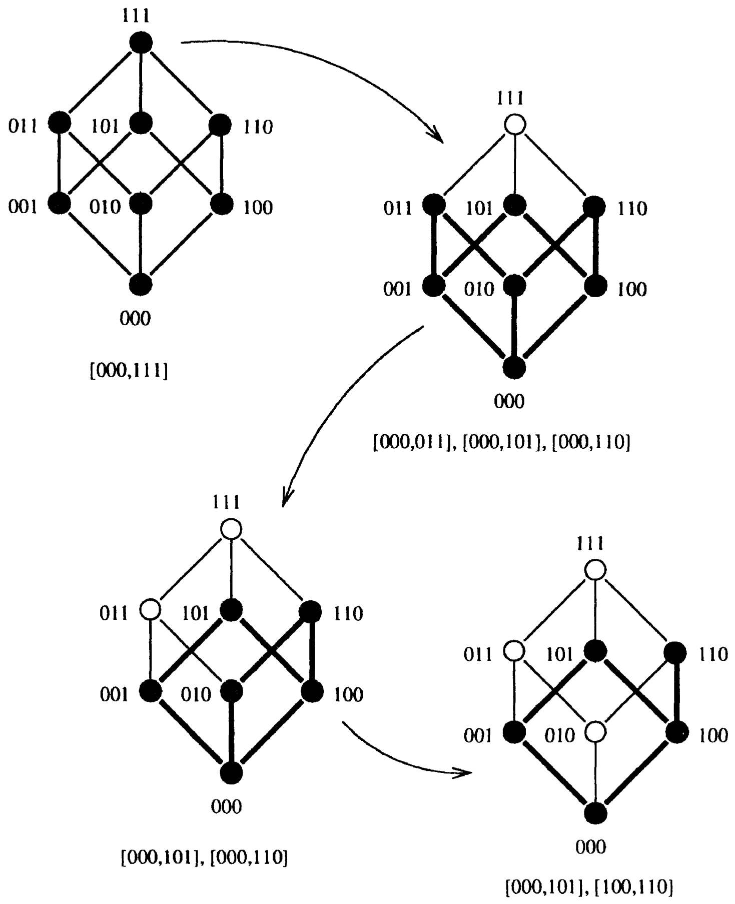


Figure 1–7. Visualization of 3-space process of the ISI algorithm.

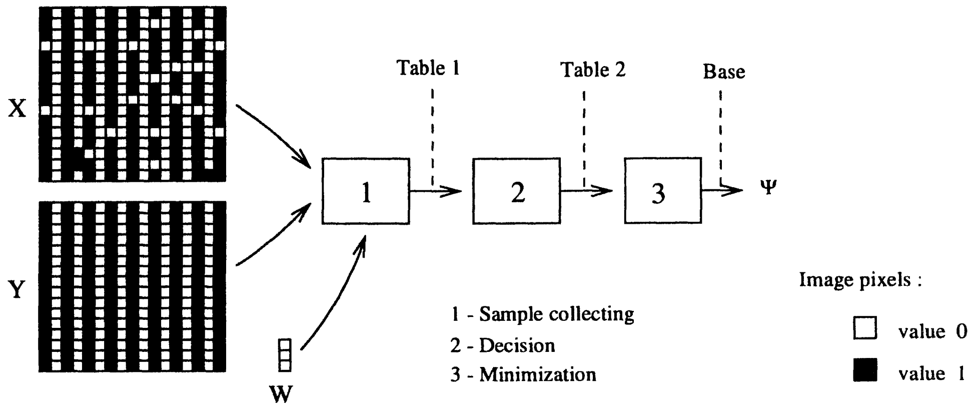


Figure 1–8. Design process.

Table 1.2. Observed statistics.

$x_1x_2x_3$	0	1
000	108	0
001	2	0
011	1	18
101	0	19
110	1	18
111	0	71

Table 1.3. Designed filter.

$x_1x_2x_3$	$f(x_1x_2x_3)$
000	0
001	0
011	1
101	1
110	1
111	1
100	$\times$
010	$\times$

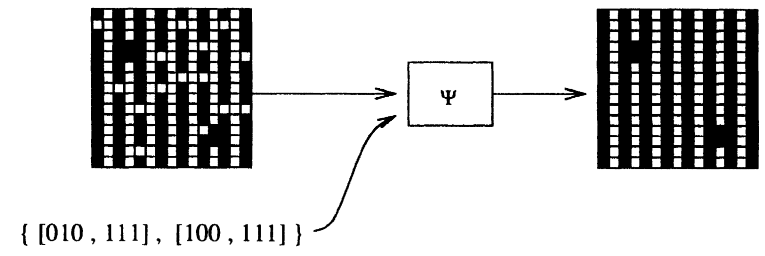


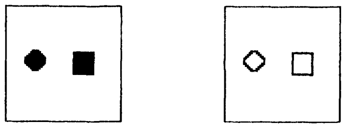
Figure 1–9. Application of designed operator.

EXAMPLE 1–1. We design a 3-variable operator to minimize additive and subtractive point noise in an image composed of vertical stripes. Figure 1–8 shows realizations of the observed and ideal images, along with a diagram with all the stages composing the design process. Table 1.2 shows the statistics obtained from the data of the figure and Table 1.3 shows the result of the decision under the MAE loss function from the data of Table 1.2. Observe that Table 1.3 defines a class of four operators that are statistically equivalent. These are obtained by specifying the don't cares (denoted by $\times$) as 0 or 1. Between these four operators we have chosen one of minimal computational cost. Figure 1–9 shows the basis of the designed operator and application of the operator to a new realization.

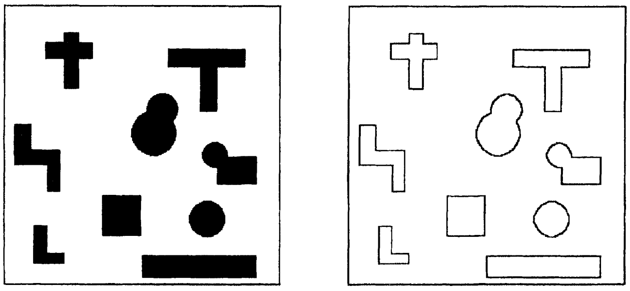
EXAMPLE 1–2 (*edge detection*). The window is the 3×3 square and the training sample is taken from the images of Fig. 1–10(a). The training sample size was 3,844. The number of distinct observed examples was 46: 24 positives and 22 negatives. Learning time was 1s. The error measure was zero. The resulting basis is composed of the following maximal intervals:

$$\begin{bmatrix} 0 & \times & \times \\ \times & 1 & \times \\ \times & \times & \times \end{bmatrix}, \begin{bmatrix} \times & \times & 0 \\ \times & 1 & \times \\ \times & \times & \times \end{bmatrix}, \begin{bmatrix} \times & \times & \times \\ \times & 1 & \times \\ \times & \times & 0 \end{bmatrix}, \begin{bmatrix} \times & \times & \times \\ \times & 1 & \times \\ 0 & \times & \times \end{bmatrix}.$$

Figure 1–10(b) shows application of the learned operator.

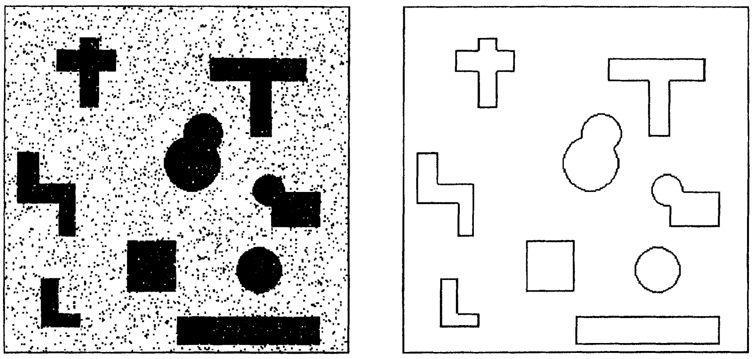


(a)

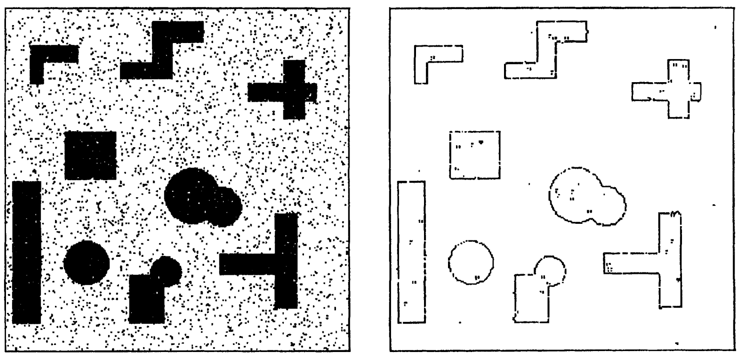


(b)

Figure 1–10. Edge detection: (a) training images; (b) application of learned operator.



(a)



(b)

Figure 1–11. Edge detection in noise: (a) training images; (b) application of learned operator.

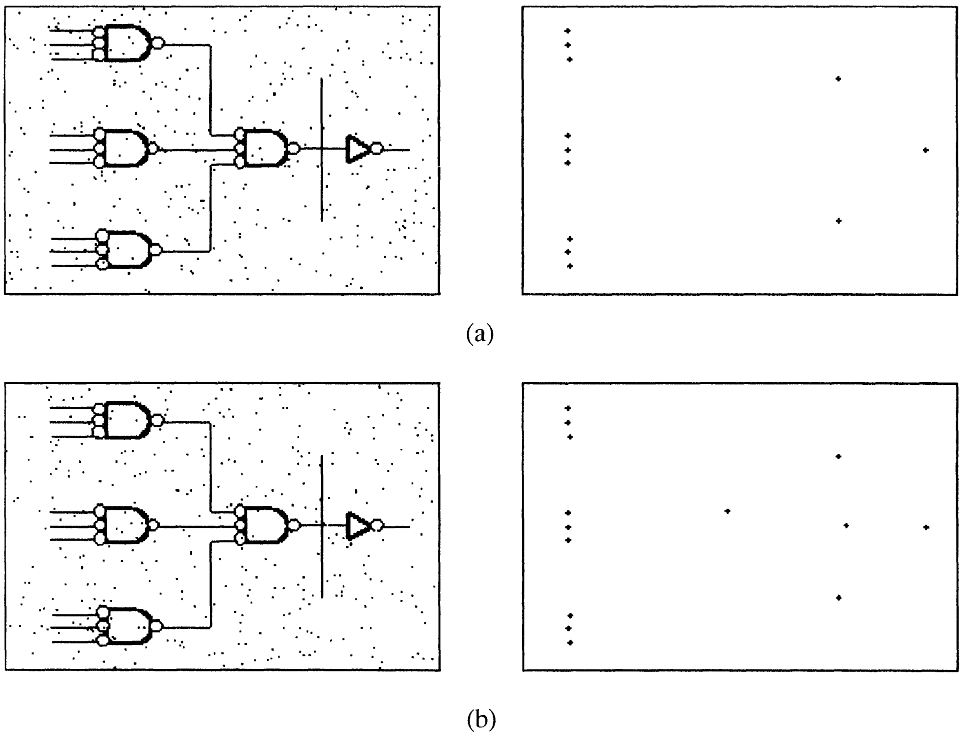


Figure 1–12. End-point detection in noise: (a) training images; (b) application of learned operator.

EXAMPLE 1–3 (*edge detection in noise*). The window is the 3×3 square. The training sample of size 80,392 was taken from the images of Fig. 1–11(a). There were 363 distinct examples observed: 278 negative and 85 positive. Learning time was 1s. The resulting basis is composed of 44 intervals. The noise is punctual, additive and subtractive, and uniformly distributed, with density 5%. The error measure was 0.48%. Figure 1–11(b) shows application of the learned operator.

EXAMPLE 1–4 (*end-point detection in noise*). The window is

$$W = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix},$$

where the 0s in the corners mean that these pixels are not part of the window. The training sample of size 36,290 was taken from two image pairs, one being shown in Fig. 1–12(a). There were 1,545 distinct observed examples: 1,538 negative and

7 positive. Learning time was 1s. The resulting basis is composed of 4 intervals:

$$\begin{pmatrix} \times & 0 & 1 & 0 & \times \\ 0 & \times & 1 & 0 & 0 \\ 0 & \times & 1 & 0 & \times \\ \times & 0 & 0 & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix}, \quad \begin{pmatrix} \times & 0 & 0 & 0 & \times \\ 0 & \times & \times & 0 & \times \\ 1 & 1 & 1 & 0 & \times \\ \times & \times & \times & 0 & \times \\ \times & \times & \times & \times & \times \end{pmatrix},$$

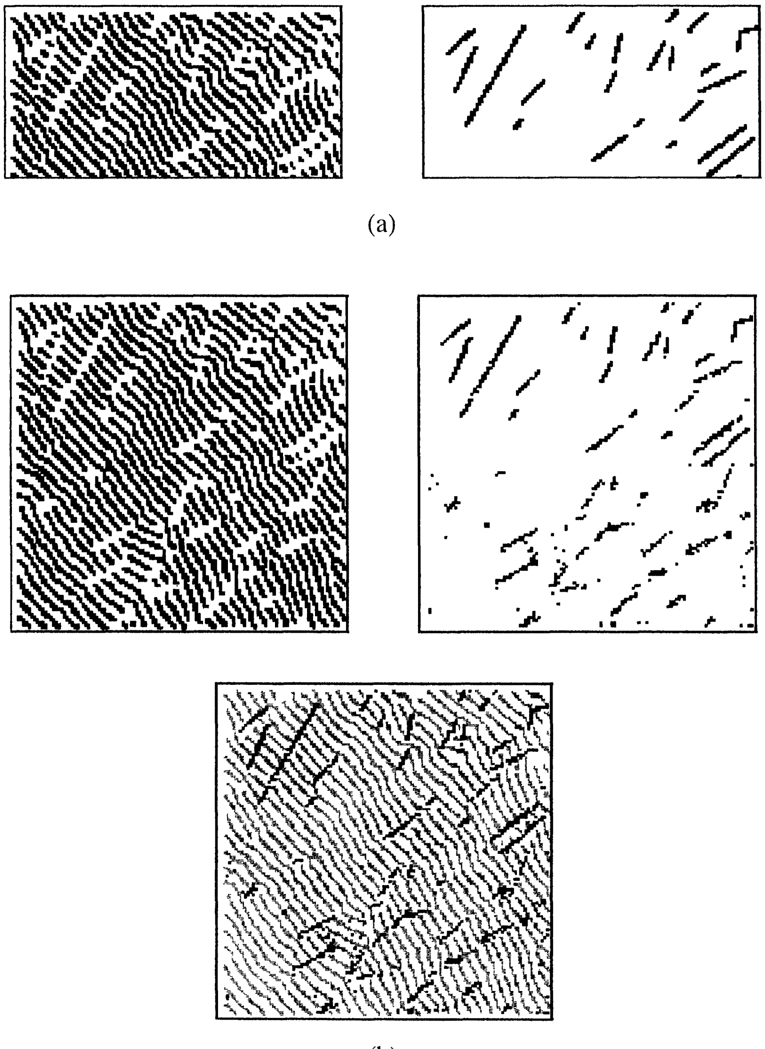
$$\begin{pmatrix} \times & \times & 0 & \times & \times \\ \times & 0 & 0 & \times & \times \\ \times & 0 & 1 & 1 & 1 \\ \times & 0 & 0 & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix}, \quad \begin{pmatrix} \times & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \times & 1 & 0 & \times \\ 0 & \times & 1 & \times & \times \\ \times & \times & 1 & \times & \times \end{pmatrix}.$$

Note that, since the corners are not part of the window, each corner pixel in the basis must be a don't-care pixel. The noise is punctual, additive and subtractive, and uniformly distributed, with densities of 1% and 0.1%. The error measure was 0.005%. Figure 1–12(b) shows application of the learned operator.

EXAMPLE 1–5 (*defect lines*). Detection of defect lines in an image of a transversal section of an eutectic alloy is a classical problem in mathematical morphology [5, 16]. To apply automatic design, first we have performed a homotopic transformation on the image and a shrink so we can observe defect lines in a low-resolution image. The transformed image has been divided into two halves, the first to be used as training data and the second for testing the result. The window is the 5×5 square. Figure 1–13(a) shows the images used for training and Fig. 1–13(b) shows application of the operator. The size of the training sample was 7,260. There were 3,812 distinct observed examples: 3,519 negatives and 293 positives. Learning time was 30s. The resulting basis is composed of 127 intervals.

1.7 CONSTRAINED OPTIMIZATION

Suboptimality often results from requiring that the chosen filter come from some subclass of all possible filters: rather than optimize over all logical expansions of the kind given in Eq. 1–1, optimize over some subclass of logical expansions. We implicitly assume that constraints are deterministic, meaning that optimization is over a filter class defined without reference to the conditional probabilities $P(Y = 1 \mid \mathbf{x})$. For unconstrained optimization, the kernel $\mathcal{K}[\psi]$ can be any subset of $\{0, 1\}^n$; for constrained optimization, there exists a family $\mathcal{Q}$ of subsets of $\{0, 1\}^n$ that is a proper subfamily of the family of all subsets of $\{0, 1\}^n$ such that $\mathcal{K}[\psi] \in \mathcal{Q}$. If ψ_{con} is the optimal logical function over the constrained filter class, then the increase in error owing to constraint is given by Eq. 1–48 with ψ_{con} in place of ψ.


(a)

(b)

Figure 1–13. Defect lines: (a) training images; (b) application of learned operator.

The error increase is zero if and only if there exists a filter in the constrained class possessing minimum MAE among all filters.

An *independent* constraint is one for which the decision whether to place a vector **x** in the kernel is constrained by a condition involving only **x** itself, and no other vectors. A number of filter conditions can result in independent constraints. Independent constraints can result from geometric conditions. Letting W be the 3×3 window and reading vectors in the usual raster method, the condition that a pixel be 1-valued if it is interior to a vertical or horizontal line of pixels produces the independent constraints

$$\begin{aligned} C_1 &: \mathbf{x} \in \mathcal{K}[\psi] \quad \text{if } \mathbf{x} \geqslant 010010010 \\ C_2 &: \mathbf{x} \in \mathcal{K}[\psi] \quad \text{if } \mathbf{x} \geqslant 000111000 \end{aligned} \tag{1–62}$$

For these independent constraints, a set $\mathcal{A}$ of vectors lies in $\mathcal{Q}$ if and only if

$$\mathcal{U}[010010010] \cup \mathcal{U}[000111000] \subset \mathcal{A} \tag{1–63}$$

A more refined constraint set occurs if the condition is changed to state that a pixel must be 1-valued if it is interior to a vertical or horizontal line of pixels and there are no other 1-valued pixels in the window. This yields the independent constraints

$$\begin{aligned} C_1 &: \mathbf{x} \in \mathcal{K}[\psi] \quad \text{if } \mathbf{x} = 010010010 \\ C_2 &: \mathbf{x} \in \mathcal{K}[\psi] \quad \text{if } \mathbf{x} = 000111000 \end{aligned} \tag{1–64}$$

For these independent constraints, a set $\mathcal{A}$ of vectors lies in $\mathcal{Q}$ if and only if

$$\{010010010, 000111000\} \subset \mathcal{A} \tag{1–65}$$

Independent constraints can arise from algebraic conditions. If we demand that Ψ be antiextensive, then there exists a single constraint:

$$C_1: \mathbf{x} \notin \mathcal{K}[\psi] \text{ if } \mathbf{x} \not\geqslant 000010000 \tag{1–66}$$

Independent constraints occur from placing a filter bound on the designed filter Ψ. For instance, $\Psi(S) \supset \Phi(S)$ $[\Psi(S) \subset \Phi(S)]$ for all S, meaning $\Psi \geqslant \Phi$ $[\Psi \leqslant \Phi]$. If Φ is increasing with $\mathcal{B}[\phi] = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, then $\Phi \leqslant \Psi$ yields m independent constraints:

$$\begin{aligned} C_1 &: \mathbf{x} \in \mathcal{K}[\psi] \quad \text{if } \mathbf{x} \geqslant \mathbf{x}_1 \\ C_2 &: \mathbf{x} \in \mathcal{K}[\psi] \quad \text{if } \mathbf{x} \geqslant \mathbf{x}_2 \\ &\vdots \\ C_m &: \mathbf{x} \in \mathcal{K}[\psi] \quad \text{if } \mathbf{x} \geqslant \mathbf{x}_m \end{aligned} \tag{1–67}$$

If Ω is increasing with $\mathcal{B}[\omega] = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$ and we desire the bounding constraint $\Phi \leqslant \Psi \leqslant \Omega$, then, in addition to the constraints $C_1, C_2, \ldots, C_m$, we have the independent constraint

$$C_{m+1}: \mathbf{x} \notin \mathcal{K}[\psi] \quad \text{if there does not exist } \mathbf{y}_j \text{ such that } \mathbf{x} \geqslant \mathbf{y}_j \tag{1–68}$$

For the bounding constraint $\Phi \leqslant \Psi \leqslant \Omega$, one can design an unconstrained estimate Ψ' of the optimal filter and then the estimate of the constrained filter is

$$\Psi = \left(\Phi \vee \Psi'\right) \wedge \Omega \tag{1–69}$$

This transformation can take place on the truth table defining Ψ', in which case it effects the following changes: if $\psi'(\mathbf{x}) = 0$ and $\phi(\mathbf{x}) = 1$, then $\psi(\mathbf{x}) = 1$; if $\psi'(\mathbf{x}) = 1$ and $\omega(\mathbf{x}) = 0$, then $\psi(\mathbf{x}) = 0$; if $\psi'(\mathbf{x})$ is undetermined (because $\mathbf{x}$ has not been seen in training), then $\psi(\mathbf{x})$ can be arbitrarily chosen so long as $\phi(\mathbf{x}) \leqslant \psi(\mathbf{x}) \leqslant \omega(\mathbf{x})$. The ISI algorithm can then be employed. Owing to the constraints, we do not concern ourselves with any conditional probability $P(Y = 1 \mid \mathbf{x})$ for which $\phi(\mathbf{x}) = 1$ or $\omega(\mathbf{x}) = 0$.

To illustrate filter-bound constraint we use opening, an increasing morphological filter to be discussed in Chapter 3. Given set A, the *opening* of set S by structuring element A, $\Gamma_A(S)$, is defined as the union of all translates of A that are subsets of S. If W is a 3×3 window and A is a 4-pixel square, then the basis of γ_A (the Boolean function for Γ_A) is

$$\mathcal{B}[\gamma_A] = \{110110000, 011011000, 000110110, 000011011\} \tag{1–70}$$

For the bounding constraint $\Gamma_A \leqslant \Psi$, we need not concern ourselves with any conditional probability $P(Y = 1 \mid \mathbf{x})$ for $\mathbf{x} \in \mathcal{U}[\mathcal{B}[\gamma_A]]$.

The error increase resulting from an independent constraint can be computed from Eq. 1–48 by letting $\mathcal{A}_0^{con}$ and $\mathcal{A}_1^{con}$ be the sets of vectors deterministically constrained to the 0-set and 1-set (kernel), respectively:

$$Er\langle\Psi_{con}\rangle - Er\langle\Psi_{opt}\rangle = \sum_{\mathbf{x} \in (\mathcal{A}_0^{con} - \mathcal{S}_0[\psi_{opt}]) \cup (\mathcal{A}_1^{con} - \mathcal{S}_1[\psi_{opt}])} \left|Ad_l(\mathbf{x})\right| \tag{1–71}$$

For the order constraints of Eq. 1–67, $\mathcal{A}_0^{con} = \emptyset$ and $\mathcal{A}_1^{con} = \mathcal{U}[\mathrm{Bas}[\phi]]$.

Independent constraints can reduce design complexity. If prior probabilities for a class C of vectors are negligible, then vectors in C can be arbitrarily independently constrained with $Er\langle\Psi_{con}\rangle - Er\langle\Psi_{opt}\rangle \leqslant \beta$, where β is the sum of the prior probabilities in C. In fact, $Er\langle\Psi_{con}\rangle - Er\langle\Psi_{opt}\rangle$ is likely to be much less than β since many vectors may be correctly constrained and the error factor $|1 - 2P(Y = 1 \mid \mathbf{x})|$ may often be substantially less than 1. Should $Er\langle\Psi_{con}\rangle - Er\langle\Psi_{opt}\rangle$ be small, the designed constrained filter can outperform the designed unconstrained filter because the difference in estimation errors between the constrained and unconstrained filters can exceed $Er\langle\Psi_{con}\rangle - Er\langle\Psi_{opt}\rangle$.

So far we have considered independent constraints; a *dependent* constraint is one that cannot be applied to each vector independently. This means there are required

Table 1.4. Weighted medians.

x	$\mathbf{a}_1$	$\mathbf{a}_2$	$\mathbf{a}_3$	$\mathbf{a}_4$
0 0 0 0	0	0	0	0
0 0 0 1	0	0	0	0
0 0 1 0	0	0	0	0
0 0 1 1	0	0	1	1
0 1 0 0	0	0	0	0
0 1 0 1	0	1	0	1
0 1 1 0	0	1	1	0
0 1 1 1	1	1	1	1
1 0 0 0	0	0	0	0
1 0 0 1	1	0	0	1
1 0 1 0	1	0	1	0
1 0 1 1	1	1	1	1
1 1 0 0	1	1	0	0
1 1 0 1	1	1	1	1
1 1 1 0	1	1	1	1
1 1 1 1	1	1	1	1

relations among vectors that do not reduce to independent constraints. A simple constraint is that there are two vectors **x** and **y** such that $\psi(\mathbf{x}) \vee \psi(\mathbf{y}) = 1$. Equivalently, $\mathbf{x} \in \mathcal{K}[\psi]$ or $\mathbf{y} \in \mathcal{K}[\psi]$. Typically, dependent constraints result from requiring that the designed filter belong to a given filter class (and it may not be easy to deduce the dependency relations).

As an illustration, consider the weighted median, which, for the binary values $x_1, x_2, \ldots, x_n$ with positive-integer weights $a_1, a_2, \ldots, a_n$, is defined by

$$\mu(x_1, x_2, \ldots, x_n) = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} a_i x_i \geqslant \left(\sum_{i=1}^{n} a_i\right)/2 \\ 0, & \text{otherwise} \end{cases} \tag{1–72}$$

Constrained optimality occurs when we desire the optimal weighted median with sum of weights a (which we assume to be odd). Consider the four-point weighted median with the sum of the weights being 5. There are four possible weight vectors: $\mathbf{a}_1 = (2, 1, 1, 1)$, $\mathbf{a}_2 = (1, 2, 1, 1)$, $\mathbf{a}_3 = (1, 1, 2, 1)$, and $\mathbf{a}_4 = (1, 1, 1, 2)$. These lead to four possible filters defined by the following minimal Boolean functions (basis expansions):

$$\begin{aligned} \mu_1(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3x_4 \\ \mu_2(x_1, x_2, x_3, x_4) &= x_1x_2 + x_2x_3 + x_2x_4 + x_1x_3x_4 \\ \mu_3(x_1, x_2, x_3, x_4) &= x_1x_3 + x_2x_3 + x_3x_4 + x_1x_2x_4 \\ \mu_4(x_1, x_2, x_3, x_4) &= x_1x_4 + x_2x_4 + x_3x_4 + x_1x_2x_3 \end{aligned} \tag{1–73}$$

The four filters are defined by Table 1.4. Having found all MAEs, filter errors can be found from the table and the optimal filter in the class is the one possessing minimum MAE. The increase in error owing to suboptimality is given by Eq. 1–48. Here it is easy to find the optimal constrained filter because only four filters must be checked. More generally, the number of filters in a given subclass can be enormous.

1.8 Optimal Increasing Filters

The most studied dependent constraint is that the filter be increasing [17–20]. The relationship among the vectors is given by: if $\mathbf{x} \leqslant \mathbf{y}$ and $\mathbf{x} \in \mathcal{K}[\psi]$, then $\mathbf{y} \in \mathcal{K}[\psi]$. If $\Im_{inc}$ is the class of all increasing operators, then the optimal increasing filter, Ψ_{inc}, is the operator in $\Im_{inc}$ possessing minimal MAE. If, for a particular model, Ψ_{opt} is increasing, then $\Psi_{inc} = \Psi_{opt}$ and reduction of the logical expansion for ψ_{opt} yields a positive expansion for ψ_{inc}. From a purely probabilistic standpoint, derivation of an increasing optimal filter via the conditional expectation and logic reduction is a valid approach. Nonetheless, given prior knowledge that Ψ_{opt} is increasing, it can be beneficial to directly design an increasing filter; that is, design Ψ_{opt} by means of a procedure that finds the optimal increasing filter, in this case that being Ψ_{opt}, itself. Furthermore, there are reasons why it might be beneficial to design the best increasing filter even when it is not fully optimal (when $\Psi_{inc} \neq \Psi_{opt}$): (1) minimal positive representations can significiantly reduce the logic cost in comparison to nonpositive representations; (2) most important statistically, typically many fewer realizations need be observed during training to obtain good estimates of optimal increasing filters compared to the number of realizations required for equivalently good estimates of nonincreasing filters. The present section briefly describes direct design of optimal MAE increasing filters. One way to proceed is to apply a switching algorithm that derives Ψ_{inc} from Ψ_{opt} by switching vectors between $\mathcal{S}_0[\psi_{opt}]$ and $\mathcal{S}_1[\psi_{opt}]$ to obtain an increasing filter for which the switching error is minimal [21, 22]. This approach has a drawback: if Ψ_{inc} cannot be obtained with a small number of switches, then switching algorithms can be prohibitively computational.

If $B \subset W$, then the single-erosion filter E_B is defined by the Boolean function

$$\varepsilon_B(\mathbf{x}) = \min\{x_i\colon b_i = 1\} \tag{1–74}$$

where $\mathbf{b} = (b_1, b_2, \ldots, b_n)$, and $b_i = 1$ if the ith pixel of W is in B and $b_i = 0$ otherwise. Filter MAE, denoted by $MAE\langle B\rangle$, is given by

$$MAE\langle B\rangle = E\big[|Y - \varepsilon_{\mathbf{b}}(\mathbf{X})|\big] = \sum_{\{(\mathbf{x},y)\colon\, y \neq \varepsilon_{\mathbf{b}}(\mathbf{x})\}} P(\mathbf{x}, y) \tag{1–75}$$

For practical design, $MAE\langle B\rangle$ is estimated from realizations of the ideal and observed images. The observed realization is eroded pixelwise and compared to the ideal-image realization. An estimate of $MAE\langle B\rangle$ is obtained by dividing the number of pixels at which the eroded observation and ideal disagree by the total number of pixels considered.

For an m-erosion filter $\Psi_{\mathcal{B}}$ with basis $\mathcal{B} = \{B_1, B_2, \ldots, B_m\}$, filter error is given by

$$\begin{aligned} MAE\langle\Psi_{\mathcal{B}}\rangle &= E\big[|Y - \psi_{\mathcal{B}}(\mathbf{X})|\big] \\ &= E\big[Y - (\varepsilon_{\mathbf{b}_1}(\mathbf{X}) \vee \varepsilon_{\mathbf{b}_2}(\mathbf{X}) \vee \cdots \vee \varepsilon_{\mathbf{b}_m}(\mathbf{X})\big] \\ &= \sum_{\{(\mathbf{x},y):\ y \neq \max_i \varepsilon_{\mathbf{b}_i}(\mathbf{x})\}} P(\mathbf{x}, y) \end{aligned} \tag{1–76}$$

An estimate of Ψ_{inc} can be found by estimating $MAE\langle\Psi_{\mathcal{B}}\rangle$ over all possible bases and choosing Ψ_{inc} as the filter corresponding to the basis generating minimal MAE. If the window has n pixels, then it has 2^n subsets, but many are eliminated from consideration owing to the minimality condition for a basis. Nonetheless, except for relatively small windows, constraints must be imposed, thereby (it is hoped slightly) increasing MAE. Constraints include limiting the basis size and constraining the search to a library (subclass) of all possible structuring elements [18]. Library constraint, requires some method of choosing the library. In practice, two techniques have dominated. *Expert libraries* are collections of structuring elements whose effects are well known (to experts) or which are basis members of popular filters known to work reasonably well for similar image models. *First-order libraries* are found by placing into the library some number of structuring elements possessing the smallest MAEs as single-erosion filters.

As expressed in Eq. 1–76, it would appear that filter design must include obtaining realization-based statistics for every basis, a prohibitive task. In fact, one need only obtain MAE estimates for single-erosion filters and then recursively obtain MAE estimates for multiple-erosion filters. According to the *morphological MAE theorem* [19], the MAE of an m-erosion filter Ψ_m can be expressed in terms of a single-erosion filter with structuring element B_m and two $(m-1)$-erosion filters Ψ_{m-1} and Φ_{m-1}:

$$MAE\langle\Psi_m\rangle = MAE\langle\Psi_{m-1}\rangle - MAE\langle\Phi_{m-1}\rangle + MAE\langle B_m\rangle \tag{1–77}$$

where the bases are given by

$$\begin{aligned} &\mathcal{B}[\Psi_{m-1}] = \{B_1, B_2, \ldots, B_{m-1}\} \\ &\mathcal{B}[\Psi_m] = \mathcal{B}[\Psi_{m-1}] \cup \{B_m\} = \{B_1, B_2, \ldots, B_m\} \\ &\mathcal{B}[\Phi_{m-1}] = \{B_1 \cup B_m, B_2 \cup B_m, \ldots, B_{m-1} \cup B_m\} \end{aligned} \tag{1–78}$$

To see how Eq. 1–77 can be used in filter design, suppose we wish to optimize by selecting bases from some structuring-element collection $\mathcal{C} = \{B_1, B_2, \ldots, B_q\}$.

For $p = 1, 2, \ldots, q$, let $\mathcal{C}_p$ be the closure of $\mathcal{C}$ under unions of p or less elements within $\mathcal{C}$. Then $\mathcal{C}_1 = \mathcal{C}$ and $\mathcal{C}_1 \subset \mathcal{C}_2 \subset \cdots \subset \mathcal{C}_q$. If we know the MAE for each structuring element in $\mathcal{C}_q$, then we can proceed recursively. For any 2-element filter Ψ_2 with basis $\mathcal{B}[\Psi_2] = \{B_{i1}, B_{i2}\}$,

$$\begin{aligned} MAE\langle\Psi_2\rangle &= MAE\langle B_{i1}, B_{i2}\rangle \\ &= MAE\langle B_{i1}\rangle + MAE\langle B_{i2}\rangle - MAE\langle B_{i1} \cup B_{i2}\rangle \end{aligned} \tag{1–79}$$

If Ψ_3 is a 3-erosion filter with basis $\mathcal{B}[\Psi_3] = \{B_{i1}, B_{i2}, B_{i3}\}$, then

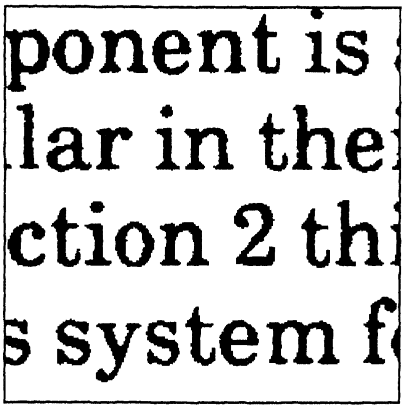
$$\begin{aligned} MAE\langle\Psi_3\rangle &= MAE\langle B_{i1}, B_{i2}, B_{i3}\rangle \\ &= MAE\langle B_{i1}, B_{i2}\rangle + MAE\langle B_{i3}\rangle \\ &\quad - MAE\langle B_{i1} \cup B_{i3}, B_{i2} \cup B_{i3}\rangle \end{aligned} \tag{1–80}$$

All terms on the right-hand side of the equation can be obtained from the previous stage of the iteration. The MAE theorem allows for evaluation of redundant structuring-element combinations but reduces to nonredundant forms.

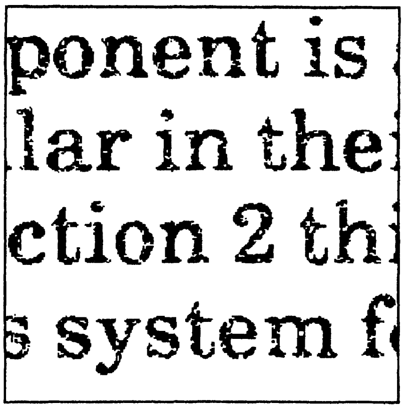
Basis-size constraint, library constraint, and estimation of filter MAEs affect the relationship between the optimal increasing filter and the designed filter meant to estimate it. If we simply impose basis-size constraint, say k structuring elements, then optimization is over the class of increasing operators having at most k terms in their minimal representations, thereby yielding a suboptimal increasing filter $\Psi_{inc}^{(k)}$. If there is also first-order-library constraint using the r structuring elements possessing minimum single-erosion MAE, then there is further suboptimality yielding a filter $\Psi_{inc}^{(k,r)}$ and

$$MAE\langle\Psi_{inc}\rangle \leqslant MAE\big\langle\Psi_{inc}^{(k)}\big\rangle \leqslant MAE\big\langle\Psi_{inc}^{(k,r)}\big\rangle \tag{1–81}$$

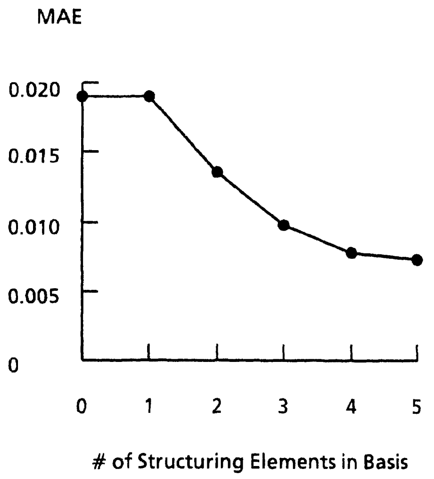
Since the designed filter is based on estimated MAEs, even without basis-size or library constraints, it is an estimate $\widehat{\Psi}_{inc}$ of Ψ_{inc}, and $MAE\langle\widehat{\Psi}_{inc}\rangle \geqslant MAE\langle\Psi_{inc}\rangle$. According to theory [12], if the number of observations is large, then $MAE\langle\widehat{\Psi}_{inc}\rangle \approx MAE\langle\Psi_{inc}\rangle$; from experience, a single 1024×1024 realization is usually sufficient for good estimation. If both basis-size and library constraint are employed, then we actually estimate $\Psi_{inc}^{(k,r)}$ from data, so that the designed filter is a statistical estimate $\widehat{\Psi}_{inc}^{(k,r)}$ of $\Psi_{inc}^{(k,r)}$, and $MAE\langle\widehat{\Psi}_{inc}^{(k,r)}\rangle \geqslant MAE\langle\Psi_{inc}^{(k,r)}\rangle$. Experience has shown that basis-size- and library-constrained estimates are close to optimal in many situations. Therefore we consider $\widehat{\Psi}_{inc}^{(k,r)}$ to be a reasonably good estimate of Ψ_{inc} and when we speak of the designed filter we are referring to $\widehat{\Psi}_{inc}^{(k,r)}$. Finally, estimation of Ψ_{inc} typically requires far less data than equivalently good estimation of Ψ_{opt}. Owing to error of estimation, $\widehat{\Psi}_{inc}^{(k,r)}$ can outperform $\widehat{\Psi}_{opt}$ even when the optimal filter is not increasing.



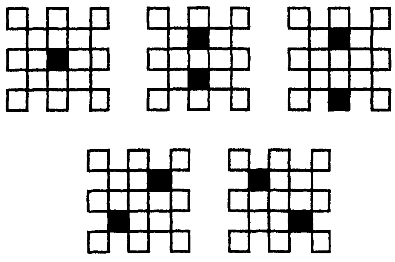
(a)



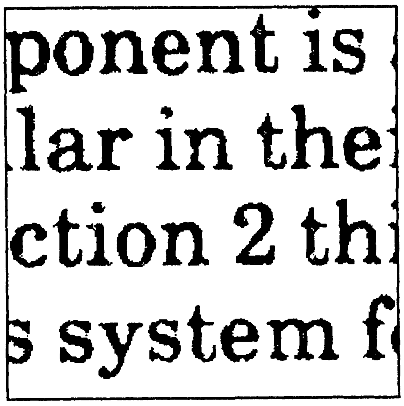
(b)



(c)



(d)



(e)

Figure 1–14. Text restoration: (a) realization of ideal process; (b) realization of degraded process; (c) MAE-vs-basis-size curve; (d) optimal 5-erosion basis; (e) restored image.

EXAMPLE 1–6 (*text restoration* [10]). Figure 1–14(a) shows a realization of an ideal text process and Fig. 1–14(b) shows a realization of the degraded process that results from pixel dropouts and for which $MAE = 0.0190$. A 300-element first-order library over a 17-pixel window has been employed in designing an increasing filter. The MAE vs. basis-size curve, optimal 5-erosion basis, and restored image are shown in Figs. 1–14(c), 1–14(d), and 1–14(e), respectively. MAE has been reduced to 0.0074 and the characters have been reconnected without joining characters.

1.9 ITERATIVE FILTERS

Another important dependent constraint is that the filter possess an iterative decomposition. Iterative filter design involves finding a statistically optimal filter for a small observation window that minimizes the error between the desired ideal image and the filtered observed image, finding a second filter that minimizes the error between the desired ideal image and the filtered output from the first filter, and then cascading the two filters to obtain a filter whose effective window is the dilation of the original small window with itself [18, 23]. The class of composite filters that can be constructed by such cascading is much smaller than the class of all filters over the large window. Hence, relative to direct optimization over the large window, an iteratively designed filter is suboptimal. But computation is greatly reduced by iterative design, often to the point where large-window optimization is computationally impossible whereas iterative design has no computational impediments. Moreover, iteratively designed filters often provide only marginally reduced performance than filters optimally designed over a large window. Finally, by employing several iterations, it is possible to achieve better filters that take less design time than could be achieved by a single-iteration method taking greater design time. We focus on increasing operators.

The key to iterative design is composition of logical sums of products. Let Ψ_1 and Ψ_2 be increasing W-operators with respective window functions

$$\begin{aligned} \psi_1(x_1, x_2, \ldots, x_n) &= \sum_k x_{k,1}x_{k,2}\cdots x_{k,m(k)} \\ \psi_2(x_1, x_2, \ldots, x_n) &= \sum_i x_{i,1}x_{i,2}\cdots x_{i,n(i)} \end{aligned} \tag{1–82}$$

The *iterative (composite) filter* $\Psi^2 = \Psi_2\Psi_1$ is defined by $\Psi_2\Psi_1(S) = \Psi_2(\Psi_1(S))$. Owing to translation invariance, we need only examine $\Psi_2\Psi_1(S)$ at 0 to arrive at the form of the window function, ψ^2, for Ψ^2. Specifically, we consider

$$\Psi_2\Psi_1(S)(0) = \psi_2\big(\psi_1(S \cap W_{x_1}), \psi_1(S \cap W_{x_2}), \ldots, \psi_1(S \cap W_{x_m})\big) \tag{1–83}$$

The functional expression on the right defines ψ^2 and depends on values of S in

$$\bigcup_{l=1}^{m} W_{x_i} = W \oplus W \tag{1–84}$$

which is the Minkowski sum of W with itself. Denoting the varibles in $W \oplus W$ by $z_1, z_2, \ldots, z_N$ and applying the representations of Eq. 1–82 to ψ^2 yields

$$\begin{aligned}\psi^2(z_1, z_2, \ldots, z_N) &= \sum_i \psi_1(S \cap W_{x_{i,1}})\psi_1(S \cap W_{x_{i,2}}) \cdots \psi_1(S \cap W_{x_{i,n(i)}})\\ &= \sum_i \left(\sum_k x_{i,1,k,1}x_{i,1,k,2} \cdots x_{i,1,k,m(k)}\right)\\ &\quad \times \left(\sum_k x_{i,2,k,1}x_{i,2,k,2} \cdots x_{i,2,k,m(k)}\right)\\ &\quad \cdots \times \left(\sum_k x_{i,n(i),k,1}x_{i,n(i),k,2} \cdots x_{i,n(i),k,m(k)}\right)\end{aligned} \tag{1–85}$$

where the variables of $S \cap W_{x_{i,j}}$ appearing in the kth product forming $\psi_1(S \cap W_{x_{i,j}})$ are $x_{i,j,k,1}, x_{i,j,k,2}, \ldots, x_{i,j,k,m(k)}$ for $j = 1, 2, \ldots, n(i)$. The resulting sum of products over $W \oplus W$ defines the window function for the iterative filter $\Psi^2 = \Psi_2\Psi_1$. Logic reduction yields product terms corresponding to $\mathcal{B}[\psi^2]$. Since the product terms for ψ_1 and ψ_2 correspond to $\mathcal{B}[\psi_1]$ and $\mathcal{B}[\Psi_2]$, respectively, derivation of $\mathcal{B}[\psi^2]$ from $\mathcal{B}[\psi_1]$ and $\mathcal{B}[\psi_2]$ is automatically accomplished via logic software.

If we desire an optimal MAE increasing filter over window $W \oplus W$ but window size makes the problem too computationally intensive, then one way to proceed is to find an optimal iterative filter. Suppose **S** is the observed image process and **I** is the ideal. An optimal increasing W-filter Ψ_1 is found to minimize MAE for $\Psi_1(\mathbf{S})$ as an estimator of **I**. Next, an optimal increasing W-filter Ψ_2 is found to minimize MAE for $\Psi_2(\Psi_1(\mathbf{S}))$ as an estimator of **I**. Relative to Ψ_2, $\Psi_1(\mathbf{S})$ is the observed image. This *optimal iterative filter* $\Psi_2\Psi_1$ is an increasing $(W \oplus W)$-filter. $\Psi_2\Psi_1$ is very likely not optimal over all $(W \oplus W)$-filters, since this would require that $\Psi_2\Psi_1$ be decomposable into W-filters; nonetheless, since filter design is computationally limited to relatively small windows, if iteration provides good suboptimal results, then it permits automatic design for larger windows than could be accomplished by direct noniterative design.

More generally, we can consider an iteration of n filters,

$$\Psi^n = \Psi_n\Psi_{n-1}\Psi_{n-2} \cdots \Psi_2\Psi_1 \tag{1–86}$$

Recursively, $\Psi^n = \Psi_n \Psi^{n-1}$. If each filter composing Ψ^n is an increasing W-filter, then Ψ^n is an increasing nW-filter, where

$$nW = W^1 \oplus W^2 \oplus \cdots \oplus W^n \tag{1–87}$$

and $W^i = W$ for $i = 1, 2, \ldots, n$. If we optimize recursively to obtain Ψ^n, then Ψ^n is an nW-filter that is suboptimal relative to the optimal filter, $\Psi_{opt,n}$, over nW. Since Ψ_k can be the identity filter, it must be that $MAE\langle\Psi^k\rangle \leqslant MAE\langle\Psi^{k-1}\rangle$. Hence,

$$MAE\langle\Psi_{opt,n}\rangle \leqslant MAE\langle\Psi^n\rangle \leqslant MAE\big\langle\Psi^{n-1}\big\rangle \leqslant \cdots \leqslant MAE\big\langle\Psi^1\big\rangle \tag{1–88}$$

Because the MAEs for the iterative filters form a decreasing sequence, they must possess a limit; however, the inequality of Eq. 1–88 does not imply that this limiting MAE is equal to $MAE\langle\Psi_{opt,n}\rangle$ for some n. Moreover, unless an optimal W^n-filter is decomposable, the leftmost inequality is strict for each n. Indeed, a basic problem of iteration is to determine the number of iterations necessary for further iterations to produce neglible improvement in MAE. Both the degree of optimality and the number of iterations necessary to be close to minimal iteratively achievable MAE are dependent on the window and the signal-noise model.

A measure of difference is needed to compare Ψ^n and $\Psi_{opt,n}$. From a purely logical perspective, the extent by which two operators disagree can be measured by the size of their switching set relative to the total number of observation vectors. The *logical difference* between operators Ψ and Φ is defined by $Card(\mathcal{Z}[\Psi, \Phi])/2^n$.

From the standpoint of filtering random sets, logical difference is not the key issue. If $\psi(\mathbf{x}) \neq \phi(\mathbf{x})$, but the probability of observing $\mathbf{x}$ is very small relative to other observation probabilities, then it matters little that $\psi(\mathbf{x}) \neq \phi(\mathbf{x})$. Their *probabilistic difference* is defined by $P(\mathcal{Z}[\Psi, \Phi])$. Applying Eq. 1–28 with the MAE loss function yields

$$\begin{aligned} MAE\langle\Psi\rangle - MAE\langle\Phi\rangle &= \sum_{\mathbf{x}\in\mathcal{S}_1[\psi]} P(\mathbf{x})P(Y=0\,|\,\mathbf{x}) + \sum_{\mathbf{x}\in\mathcal{S}_0[\Psi]} P(\mathbf{x})P(Y=1\,|\,\mathbf{x}) \\ &\quad - \sum_{\mathbf{x}\in\mathcal{S}_1[\phi]} P(\mathbf{x})P(Y=0\,|\,\mathbf{x}) - \sum_{\mathbf{x}\in\mathcal{S}_0[\phi]} P(\mathbf{x})P(Y=1\,|\,\mathbf{x}) \\ &= \sum_{\mathbf{x}\in\mathcal{S}_1[\psi]-\mathcal{S}_1[\phi]} P(\mathbf{x})\big[P(Y=0\,|\,\mathbf{x}) - P(Y=1\,|\,\mathbf{x})\big] \\ &\quad + \sum_{\mathbf{x}\in\mathcal{S}_1[\phi]-\mathcal{S}_1[\psi]} P(\mathbf{x})\big[P(Y=1\,|\,\mathbf{x}) - P(Y=0\,|\,\mathbf{x})\big] \end{aligned} \tag{1–89}$$

Each term in both of the latter sums is bounded by $P(\mathbf{x})$, so that

$$\big|MAE\langle\Psi\rangle - MAE\langle\Phi\rangle\big| \leqslant P\big(\mathcal{Z}[\Psi, \Phi]\big) \tag{1–90}$$

and the probabilistic difference between two filters serves as an upper bound on the difference between their MAEs.

In designing iterative filters, it is not unusual for the designed estimates of Ψ^n and $\Psi_{opt,n}$ to be substantially different while their probabilistic difference is negligible. Hence, they differ significantly as logical operators but insignficantly as filters on the observed random set.

EXAMPLE 1–7 (*text restoration* [24]). To illustrate increasing-filter iterative design, we consider document restoration. Because we wish to compare optimal filtering over $W \oplus W$ with iterative filtering over W, we employ the small window

$$W = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

for which

$$W \oplus W = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

An ideal document image is subjected to dilation and random pepper noise. Iterative filters using both W and $W \oplus W$ are designed to restore the original. Using W, two iterations achieve essentially the same degree of restoration as a single application of the filter designed over $W \oplus W$. Not only is design much faster for the iterative filter, were we to employ a 5×5 iterative filter, two stages would approximate a 9×9 optimal filter, which could not be designed by the increasing-filter methodology. Figure 1–15 shows a realization of the ideal image, a degraded version of the realization, restoration after a single-stage W-filter, restoration after a two-stage W-filter, and restoration after a single-stage $(W \oplus W)$-filter.

EXAMPLE 1–8 (*connected thinning*). We desire a thinning (skeletonization) algorithm based on iterative filtering over a 3×3 window to thin connected components. Figure 1–16 shows a realization of the input process and the results of four iterations of the designed filter. Note how most of the transformation is accomplished in the first iteration and how small "corrections" are made subsequently. This is not unusual for iterative filtering, where later-stage filters can "correct over filtering" of early stages [24]. Figure 1–17 shows the realization of Fig. 1–16(a) with 5% random pepper noise and the result of the designed iterative filter after four interations. Table 1.5 gives error percentages for each iteration.

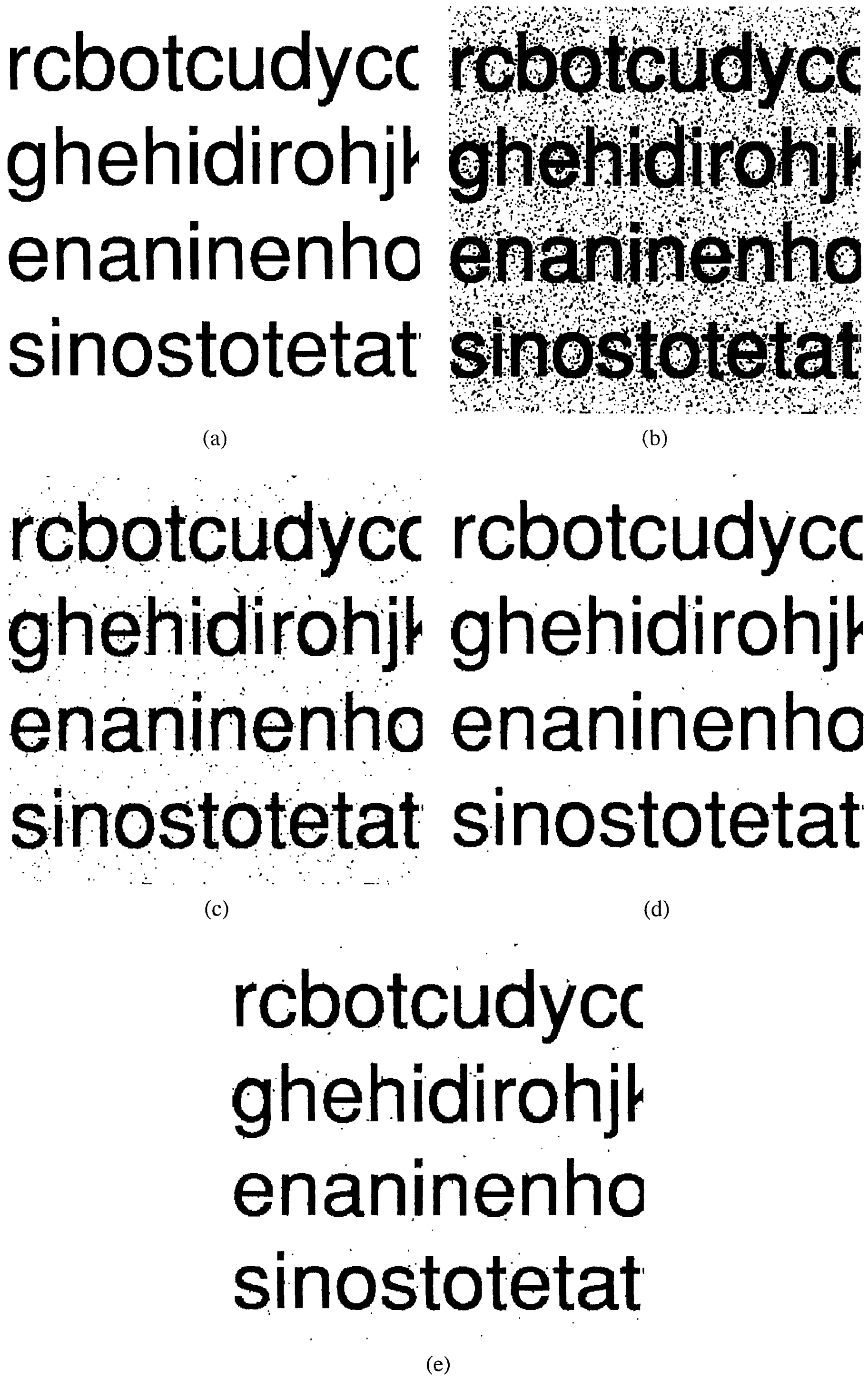


Figure 1–15. Text restoration: (a) ideal realization; (b) degraded realization; (c) restoration with single-stage W-filter; (d) restoration with two-stage W-filter; (e) restoration with single stage $(W \oplus W)$-filter.

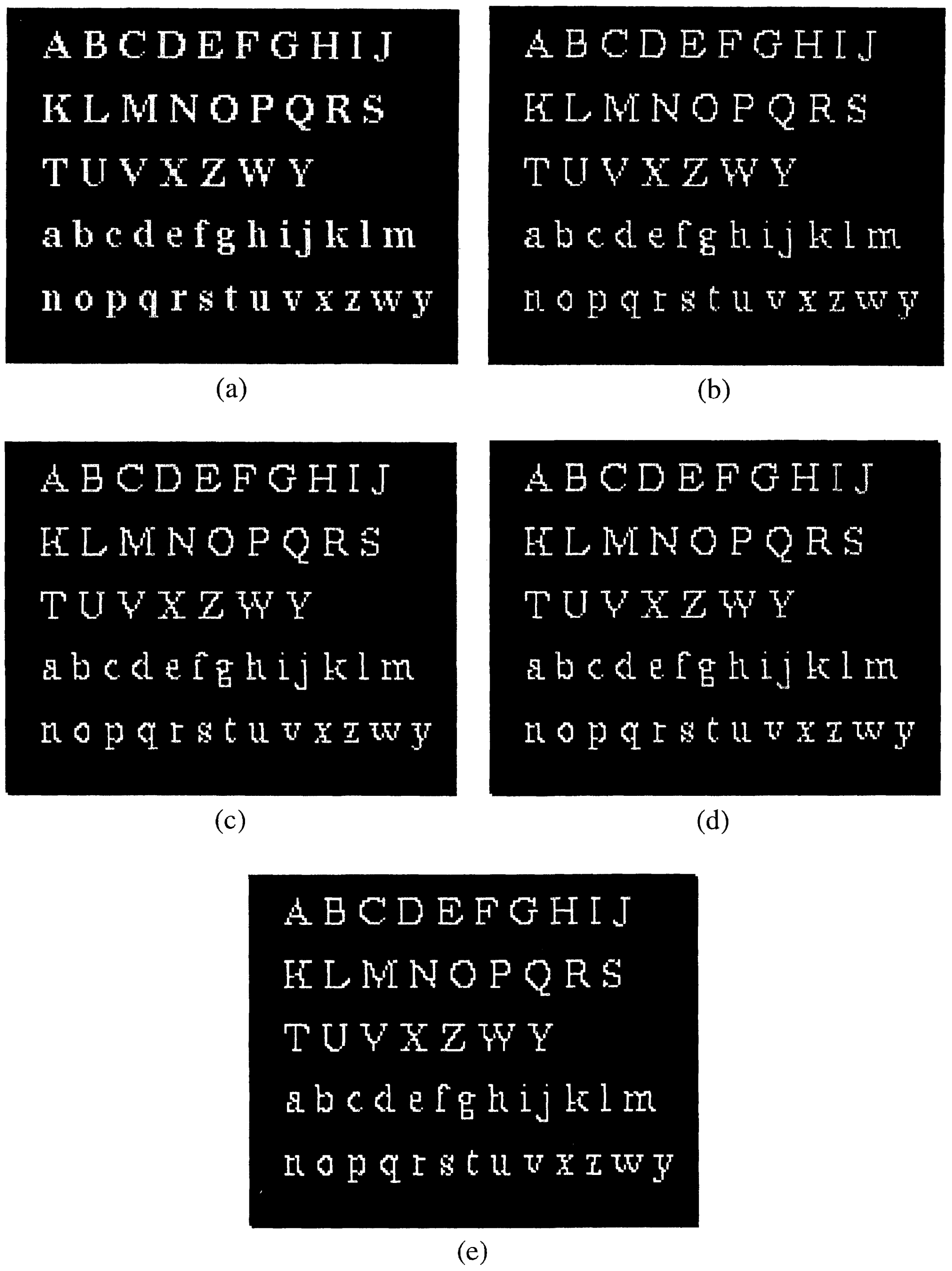
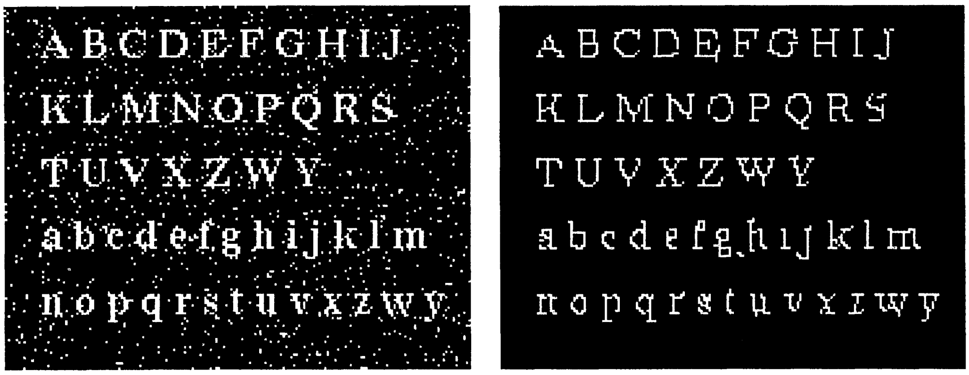


Figure 1–16. Connected thinning: (a) realization of input process; (b) result of first iteration; (c) result of second iteration; (d) result of third iteration; (e) result of fourth iteration.

Table 1.5. Error percentages for connected thinning.

Iteration	Nonnoisy	Noisy
1	0.40%	1.20%
2	0.20%	0.96%
3	0.18%	0.88%
4	0.17%	0.83%



(a) (b)

Figure 1–17. Realization of Fig. 1–16(a) with 5% random pepper noise and the result of the designed iterative filter after four interations.

EXAMPLE 1–9 (*character recognition* [15]). The present example combines algebraic constraint and iterative filtering for character recognition. It has been performed on an Intel-586 processor and processing time is measured in hours (h), minutes (m), and seconds (s). The input image is text and the output a set of markers for a desired character. Operators are constrained to be antiextensive [meaning $S \supset \Psi(S)$], the first training stage uses the SR loss function, subsequent stages use the MAE loss function, the first filter is over either a 7×7 or 9×9 window, and each subsequent stage uses a window reduced by 2 pixels per side compared to its preceding stage. Figure 1–18 shows an input image (gray) with markers for the character 'a' superimposed. Table 1.6 provides some sample results for recognition of 'a'.

1.10 MACHINE LEARNING THEORY AND OPTIMAL OPERATOR DESIGN

Computational learning theory is one of the first attempts to construct a mathematical model for the cognitive concept-learning process. It provides a framework for studying a variety of algorithmic processes. We briefly review basic elements of

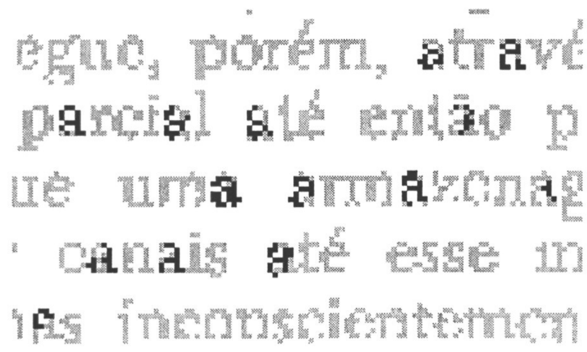


Figure 1–18. Character recognition: input gray image with markers for the character 'a' superimposed.

Table 1.6. Training data.

First training stage	# Examples (thousands)	# of Stages	Basis size	Training time	Relative error (%)
7×7	79	1	644	2 hour/47 min	10.4
7×7	88	2	726	2 hour/48 min	1.4
7×7	96	3	762	2 hour/48 min	0.5
9×9	79	1	551	3 hour/45 min	12.8
9×9	88	2	700	3 hour/48 min	0.8
9×9	96	3	760	3 hour/48 min	0.5
9×9	104	4	781	3 hour/48 min	0.4

the model and then discuss the relationship between automatic design of optimal W-operators and Haussler's paradigm for learning Boolean concepts in the context of machine learning theory [8, 25, 26].

Consider a finite set (*domain*) of objects D structured by an unknown distribution μ. A *concept* c is a subset of objects in a predefined domain D or, equivalently, a Boolean function from $D \to \{0, 1\}$. An *example* of a concept c is a pair (x, b), where x is an object in D and b is a binary *label* (0 or 1) indicating whether or not $x \in c$. If $b = 1$, then $x \in c$ and the example is called *positive*; otherwise, $x \notin c$ and the example is called *negative*. An object is taken randomly from the domain and a *teacher*, who knows the concept, classifies the object as a positive or negative example. In more sophisticated models, the teacher is assumed to be imperfect, that is, the teacher may erroneously classify some objects. In the imperfect case, the teacher is modeled by a conditional distribution $P(\mathbf{B} \mid \mathbf{X})$, where $\mathbf{X}$ is the random process, with distribution μ, representing the domain, and $\mathbf{B}$ is a binary random process labeling a domain object when it is observed.

Concept learning is the process by which a learner constructs a good approximation to an unknown concept from a number of examples and possibly some prior information about the concept to be learned. After observing labels of a sequence of objects taken randomly from the domain, the learner obtains a Boolean function that decides whether an object taken from the domain is an element of the concept with a small probability of error. The set of all possible concepts to be learned is called the *hypothesis space* and denoted by H. The concept $t \in H$ to be determined is called the *target concept*. The task is to find a concept $h \in H$, called a *hypothesis*, that is a good approximation of t.

A *training sample* **s** of length m is a sequence of m examples,

$$\mathbf{s} = \big((x_1, b_1), (x_2, b_2), \ldots, (x_m, b_m)\big) \tag{1–91}$$

where, for $i = 1, 2, \ldots, m$, x_i is an object and b_i a label. Assuming that object selection from the domain is independent, it may happen that an object occurs more than once in the examples that constitute **s**. When the teacher does not make mistakes, the training sample is said to be *consistent*; that is, if $x_i = x_j$, then $b_i = b_j$. When the teacher may err, the training sample is said to be *nonconsistent*; that is, it may happen that $b_i \neq b_j$ when $x_i = x_j$. For fixed m, the class **S** of training samples is a random process with

$$P(\mathbf{S} = \mathbf{s}) = \prod_{i=1}^{m} P(\mathbf{X} = x_i) P(\mathbf{B} = b_i \mid x_i) \tag{1–92}$$

The probability mass for **S** is induced by the probability mass on the domain in conjunction with the conditional labeling probabilities. A *learning algorithm* is a function L that assigns to any training sample **s**, for a target concept $t \in H$, a *hypothesis* $h \in H$. We write $h = L(\mathbf{s})$.

Consider a fixed target concept $t \in H$ and a given loss function l. Let **B** be the Boolean random process describing the label attributed to a domain object when it is observed in the learning procedure. For any hypothesis $h \in H$, the *risk* $r(h, t)$ of choosing hypothesis h for the concept t is defined by

$$r(h, t) = E\big[l\big(\mathbf{B}, h(\mathbf{X})\big)\big] \tag{1–93}$$

where **X** is the random process modeling the domain. Because **B** depends on **X**, the distributions of both **B** and $h(\mathbf{X})$ depend on the distribution of **X**. Let $h^* \in H$ be the hypothesis of minimum risk for t, meaning $r(h, t) \geqslant r(h^*, t)$ for all $h \in H$.

Algorithm L is called a *probably approximately correct* (*PAC*) learning algorithm for the hypothesis space H if, given real numbers ε $(0 < \varepsilon < 1)$ and δ $(0 < \delta < 1)$, there exists a positive integer $m(\varepsilon, \delta)$ such that $m \geqslant m(\varepsilon, \delta)$ implies

$$P\left(\left|r(L(\mathbf{S}), t) - r\left(h^*, t\right)\right| < \varepsilon\right) > 1 - \delta \tag{1–94}$$

for any target concept $t \in H$, distribution μ on D, and conditional distribution $P(\mathbf{B} \mid \mathbf{X})$. The value $m(\varepsilon, \delta)$ and the pair (ε, δ) are called, respectively, the *sample complexity* and *precision* of the learning algorithm. If H is finite, then the sample complexity of any PAC learning algorithm L is bounded [26] by

$$m(\varepsilon, \delta) = \frac{1}{\varepsilon^2}\left(\log Card(H) + \log \frac{1}{\delta}\right) \tag{1–95}$$

The formulation of PAC learning we have presented is a simplification of Haussler's formulation to concepts understood in the sense defined here. If the loss function is

$$l\left(\mathbf{B}, h(\mathbf{X})\right) = \left|\mathbf{B} - h(\mathbf{X})\right| \tag{1–96}$$

and the training sample for a given target concept t is consistent, then $r(h^*, t) = r(t, t) = 0$,

$$r(h, t) = \mu\left(\left\{x \in D\colon\ t(x) \neq h(x)\right\}\right) \tag{1–97}$$

the formulation reduces to the original formulation of PAC learning algorithms [27] and the bound for the sample complexity reduces to

$$m(\varepsilon, \delta) = \frac{1}{\varepsilon}\left(\log Card(H) + \log \frac{1}{\delta}\right) \tag{1–98}$$

The complete procedure of estimation of set operators we have proposed (estimation of the conditional probabilities and estimation of the best operator by optimization) is equivalent to the learning-concept formulation just presented. W-operators are equivalent to Boolean functions, and concepts (in the sense we have defined) are Boolean functions defined on a given domain. When interpreting W-operators as concepts, the domain is the set of patterns observed in W; the distribution μ gives the relative proportions of observed patterns and is determined by the probabilities $P(\mathbf{X} = \mathbf{x})$; and the conditional probability $P(\mathbf{B} \mid \mathbf{X})$ results from nondeterminacy in the ideal image given an observed pattern and noise affecting the images.

In the hit-or-miss representation, for a hypothesis (Boolean function) h (determining a W-operator Ψ), a loss occurs if and only if $(E, F) \in C_\Psi$, where (E, F) is the canonical structuring pair equivalent to $\mathbf{x}$, and $\mathbf{B} = 0$ when $\mathbf{x}$ is observed, or

$(E, F) \notin \mathcal{C}_\Psi$ and $\mathbf{B} = 1$ when $\mathbf{x}$ is observed. Since $(E, F) \in \mathcal{C}_\Psi$ can be equivalently expressed as $\mathbf{x} \in \mathcal{C}_\Psi$ and $h(\mathbf{x}) = 1$ if and only if $\mathbf{x} \in \mathcal{C}_\Psi$,

$$\begin{aligned} r(h,t) &= E\big[l(\mathbf{B}, h(\mathbf{X}))\big] \\ &= \sum_{\mathbf{x}} E\big[l(\mathbf{B}, h(\mathbf{x})) \mid \mathbf{x}\big] P(\mathbf{X}=\mathbf{x}) \qquad (1\text{–}99) \\ &= \sum_{\mathbf{x}\in\mathcal{C}_\Psi} E\big[l(\mathbf{B},1) \mid \mathbf{x}\big] P(\mathbf{X}=\mathbf{x}) + \sum_{\mathbf{x}\notin\mathcal{C}_\Psi} E\big[l(\mathbf{B},0) \mid \mathbf{x}\big] P(\mathbf{X}=\mathbf{x}) \end{aligned}$$

Assuming the target function t is a possible hypothesis,

$$\begin{aligned} r(h,t) - r(t,t) = &\sum_{\mathbf{x}\in\mathcal{C}_\Psi - \mathcal{C}_l} \Big(E\big[l(\mathbf{B},1) \mid \mathbf{x}\big] - E\big[l(\mathbf{B},0) \mid \mathbf{x}\big]\Big) P(\mathbf{X}=\mathbf{x}) \\ &+ \sum_{\mathbf{x}\notin\mathcal{C}_l - \mathcal{C}_\Psi} \Big(E\big[l(\mathbf{B},0) \mid \mathbf{x}\big] - E\big[l(\mathbf{B},1) \mid \mathbf{x}\big]\Big) P(\mathbf{X}=\mathbf{x}) \end{aligned} \qquad (1\text{–}100)$$

where $\mathcal{C}_l$ is the class of structuring pairs corresponding to the W-operator defined by the target Boolean function t. Since the $r(h, t)$ is minimized for $h = t$, according to Eq. 1–99 applied to t, $E[l(\mathbf{B}, 1) \mid \mathbf{x}] \leqslant E[l(\mathbf{B}, 0) \mid \mathbf{x}]$ if $\mathbf{x} \in \mathcal{C}_l$ and $E[l(\mathbf{B}, 0) \mid \mathbf{x}] \leqslant E[l(\mathbf{B}, 1) \mid \mathbf{x}]$ if $\mathbf{x} \notin \mathcal{C}_l$. Hence, Eq. 1–100 reduces to

$$r(h,t) - r(t,t) = \sum_{\mathbf{x}\in\mathcal{C}_\Psi \Delta \mathcal{C}_l} \Big|E\big[l(\mathbf{B},1) \mid \mathbf{x}\big] - E\big[l(\mathbf{B},0) \mid \mathbf{x}\big]\Big| P(\mathbf{X}=\mathbf{x}) \qquad (1\text{–}101)$$

which is equivalent to the suboptimality error-increase expression given in terms of absolute advantages in Eq. 1–48. Interpreting Eq. 1–48 relative to sampling, the concept-learning precision inequality of Eq. 1–94 takes the form

$$1 - \delta < P\Big(\big|Er\langle\widehat{\Psi}_l\rangle - Er\langle\Psi_l\rangle\big| < \varepsilon\Big) = P\left(\sum_{(E,F)\in\widehat{\mathcal{C}}_l \Delta \mathcal{C}_l} \big|Ad_l(E,F)\big| < \varepsilon\right) \qquad (1\text{–}102)$$

for $m \geqslant m(\varepsilon, \delta)$.

General machine-learning sample complexity bounds are unrealistically loose when compared with practical results found in the literature [9, 12, 15]. A salient reason is use of prior information, which is information that a learner has about the domain or concept to be learned. If this information is used properly, then the training-sample size needed to obtain a given precision (ε, δ) can become smaller or, equivalently, training samples of a fixed size can give sharper estimates.

A key issue in learning W-operators is the choice of a window W [18]. Window size should be as small as possible since the size of the domain D is affected exponentially by the size of W, $Card(D) = 2^{Card(W)}$. In this task, prior information plays an important role. Often knowledge of some geometrical properties expressed by the concepts is sufficient to properly choose W. For instance, if the target concept is an operator to detect edges, an edge in digital topology depends on just a small neighborhood of 4 or 8 pixels, depending on the connectivity required [28]; if the target concept is a filter of connected components or holes, then shape recognition can be described by canonical hit-or-miss operators that depend on the smallest window containing all components to be filtered [29–31].

Window size is related to image resolution: higher resolutions require larger windows. Low resolutions require less training data to achieve a given estimation precision. For instance, consider a training-data set consisiting of N pairs of square images with n lines and a square window of r lines, with $r \ll n$. There are $m = 2^{r^2}$ pattern vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ defining the full observation domain. The training data yield Nn^2 examples (observations) and, owing to repetitions, many fewer than Nn^2 of the potential m patterns are observed to cover the full observation domain of size m. If images with half the resolution are sufficient for the task, then we could employ N square images with $n/2$ lines and a square window with $r/2$ lines. In this case, we observe $Nn^2/4$ examples and the domain size is $2^{r^2/4}$. At half resolution, the number of examples is divided by 4, as is the exponent of the domain size. Choosing a minimum resolution sufficient to solve an imaging problem is a basic preliminary step.

Practical image analysis is usually restricted to specific contexts, that is, to particular classes of images for which the operators should perform well. Restriction to a given context implies that the domain D becomes a subset of the power set of W. It is typically not easy to estimate the number of patterns in a given context, but the number is often significantly smaller than $2^{Card(W)}$. The worst probabilistic structure for the domain is the uniform distribution, which models the most disorganized space. This distribution is poor for concept learning because to obtain small risks for a hypothesis h that estimates a target concept t, a large portion of the domain must be covered by examples of the training sample. Thus, very large training samples are needed. The best probability structure for the domain is the deterministic one, where there is just one pattern with probability 1. Practical applications involve neither of these extremes and are usually far from both.

Information concerning properties of the target operator is also useful. If the target operator is a marker for shape recognition, then it is antiextensive; if it is a size classifier, then it is increasing, antiextensive and idempotent; etc. These kinds of properties can be interpreted as constraints that characterize families of operators in the hypothesis space. Under knowledge of target-operator properties, the hypothesis space will be the intersection of the families of operators defined by the constraints.

Another kind of prior knowledge is a good initial hypothesis for the target operator [32]. Instead of learning the target operator directly, we can learn the symmetric difference between the target operator and the initial hypothesis. The nearer the initial hypothesis to the target operator, the easier will be the task of learning the symmetric difference between them. Use of differencing representation for document restoration is an example of this kind of prior knowledge and, for it, the identity is used as the initial hypothesis. When there is a very small amount of noise affecting just the edges of text characters, the identity operator is a reasonable initial hypothesis.

In general, learning a target concept corresponds to generating a function (the hypothesis) by the specifications of values (labels) for the most probable subsets of W. For generating the hypothesis, patterns are chosen randomly from the domain. If the number of patterns with unknown classification is large, then it will be necessary to have large training samples to get good precision; otherwise the training samples may be smaller to get the same precision. Prior information can reduce the number of subsets of W with an unknown label, thereby yielding smaller sample complexities. A small window reduces domain size. Context can determine subsets of W that should be labeled; others are taken as don't-cares. Distributional knowledge allows patterns with small probabilities to be treated as don't-cares. Operator properties facilitate deduction of some pattern labels from knowledge of other pattern labels. An initial hypothesis may label a large number of patterns.

1.11 ROBUSTNESS

A fundamental aspect of any filter is the degree to which its performance degrades when it is applied to random processes different than the one for which it has been designed. Qualitatively, a filter is said to be *robust* when its performance degradation is acceptable for processes statistically close to the design process. Robustness is crucial for application because a filter will surely be applied in nondesign settings. This may occur because it is applied to different stationary random processes or to nonstationary random processes. For instance, an optimal document filter may be applied to documents using different fonts than the design fonts. Robustness depends on both the ideal and observed images.

To define robustness, consider a parameterized ideal discrete random set $\mathbf{I_r}$, where $\mathbf{r}$ is a parameter vector determining the probability law governing $\mathbf{I_r}$, and a parameterized system transformation $\Xi_\mathbf{t}$, whose probability law is determined by the parameter $\mathbf{t}$. The observed random set is $\mathbf{S_a} = \Xi_\mathbf{t}(\mathbf{I_r})$, where $\mathbf{a} = (\mathbf{r}, \mathbf{t})$, and the filter problem is to design an optimal filter $\Psi_\mathbf{a}$ to restore $\mathbf{I_r}$ by means of the estimator $\Psi_\mathbf{a}(\mathbf{S_a})$. Taken together, $\mathbf{I_r}$, $\Xi_\mathbf{t}$, and $\mathbf{S_a}$ form a parameterized system model $\mathcal{M}_\mathbf{a}$, and $\Psi_\mathbf{a}$ is optimal relative to $\mathcal{M}_\mathbf{a}$. To unify notation, for $\mathbf{a} = (\mathbf{r}, \mathbf{t})$ we notate all aspects of the model by $\mathbf{a}$, namely, $\mathbf{I_a}$, $\Xi_\mathbf{a}$, and $\mathbf{S_a}$, where it is understood that $\mathbf{I_a}$ and $\Xi_\mathbf{a}$ are each parameterized by separate components of $\mathbf{a}$.

For a filter Ψ, the error of Eq. 1–27 must indexed by the model parameter. Let $Er_{\mathbf{a}}\langle\Psi\rangle$ denote the error for Ψ relative to the system model $\mathcal{M}_{\mathbf{a}}$. If $\Psi_{\mathbf{b}}$ is optimal relative to $\mathcal{M}_{\mathbf{b}}$, then $Er_{\mathbf{a}}\langle\Psi_{\mathbf{b}}\rangle \geqslant Er_{\mathbf{a}}\langle\Psi_{\mathbf{a}}\rangle$: $\Psi_{\mathbf{a}}$ is a better estimator of $\mathbf{I}_{\mathbf{a}}$ than is $\Psi_{\mathbf{b}}$. Intuitively, for $\Psi_{\mathbf{b}}$ to be robust, the inequality should not be too great when $\mathbf{b}$ is close to $\mathbf{a}$. Hence, robustness of the optimal filter $\Psi_{\mathbf{b}}$ for model $\mathcal{M}_{\mathbf{b}}$ relative to $\mathcal{M}_{\mathbf{a}}$ is defined by

$$\kappa(\mathbf{b};\, \mathbf{a}) = Er_{\mathbf{a}}\langle\Psi_{\mathbf{b}}\rangle - Er_{\mathbf{a}}\langle\Psi_{\mathbf{a}}\rangle \tag{1–103}$$

For the MAE loss function,

$$\kappa(\mathbf{b};\, \mathbf{a}) = E\big[|Y_{\mathbf{a}} - \psi_{\mathbf{b}}(\mathbf{X}_{\mathbf{a}})|\big] - E\big[|Y_{\mathbf{a}} - \psi_{\mathbf{a}}(\mathbf{X}_{\mathbf{a}})|\big] \tag{1–104}$$

where $Y_{\mathbf{a}}$ and $\mathbf{X}_{\mathbf{a}}$ are the ideal value and observation random vector for model $\mathcal{M}_{\mathbf{a}}$, and $\psi_{\mathbf{b}}$ and $\psi_{\mathbf{a}}$ are the optimal Boolean functions for $\mathcal{M}_{\mathbf{b}}$ and $\mathcal{M}_{\mathbf{a}}$, respectively [33]. $\kappa(\mathbf{b};\, \mathbf{a}) \geqslant 0$ and $\kappa(\mathbf{a};\, \mathbf{a}) = 0$. $\kappa(\mathbf{b};\, \mathbf{a})$ is not symmetric with respect to $\mathbf{a}$ and $\mathbf{b}$, since except in special circumstances $\kappa(\mathbf{b};\, \mathbf{a}) \neq \kappa(\mathbf{a};\, \mathbf{b})$.

Practically, we desire some degree of robustness. For robustness of a particular designed filter $\Psi_{\mathbf{b}}$, we say $\Psi_{\mathbf{b}}$ is *robust to the degree* (ε, δ) if $\kappa(\mathbf{b};\, \mathbf{a}) \leqslant \varepsilon$ when $|\mathbf{b} - \mathbf{a}| \leqslant \delta$. If the vector $(\mathbf{a}, \mathbf{b})$ can vary over some region R, then we say that the optimal filter is *uniformly robust to the degree* (ε, δ) over R if $\kappa(\mathbf{b};\, \mathbf{a}) \leqslant \varepsilon$ when $|\mathbf{b} - \mathbf{a}| \leqslant \delta$ and $(\mathbf{a}, \mathbf{b}) \in R$.

Focusing on MAE, some probabilistic calculations show that for a given vector $\mathbf{x}$, the increase in MAE owing to using the filter $\Psi_{\mathbf{b}}$ instead of $\Psi_{\mathbf{a}}$ for the system $\mathcal{M}_{\mathbf{a}}$ is

$$\kappa_{\mathbf{x}}(\mathbf{b};\, \mathbf{a}) = |2P_{\mathbf{a}}(Y = 1 \mid \mathbf{x}) - 1|\zeta_{\mathbf{x}}(\mathbf{b};\, \mathbf{a})P_{\mathbf{a}}(\mathbf{x}) \tag{1–105}$$

where $\zeta_{\mathbf{x}}(\mathbf{b};\, \mathbf{a})$ is an auxiliary function defined by

$$\zeta_{\mathbf{x}}(\mathbf{b};\, \mathbf{a}) = \begin{cases} 0, & \text{if } P_{\mathbf{b}}(Y = 1 \mid \mathbf{x}) \geqslant 0.5,\ P_{\mathbf{a}}(Y = 1 \mid \mathbf{x}) \geqslant 0.5 \\ 0, & \text{if } P_{\mathbf{b}}(Y = 1 \mid \mathbf{x}) < 0.5,\ P_{\mathbf{a}}(Y = 1 \mid \mathbf{x}) < 0.5 \\ 1, & \text{otherwise} \end{cases} \tag{1–106}$$

Robustness is obtained by summing $\kappa_{\mathbf{x}}(\mathbf{b};\, \mathbf{a})$ over all $\mathbf{x}$.

Depending on the model $\mathcal{M}_{\mathbf{a}}$, for any observation vector $\mathbf{x}$ there is a set of vectors $\mathcal{U}_{\mathbf{a},\mathbf{x}} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{m(\mathbf{x})}\}$ arising from realizations of $\mathbf{I}_{\mathbf{a}}$ via intersections with W_z such that the center pixel is 1-valued and the transition $\mathbf{u}_i \to \mathbf{x}$ is possible with application of $\Xi_{\mathbf{a}}$. There also is a collection $\mathcal{V}_{\mathbf{a},\mathbf{x}} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{n(\mathbf{x})}\}$ such that the center pixel is 0-valued and the transition $\mathbf{v}_j \to \mathbf{x}$ can occur under $\Xi_{\mathbf{a}}$. Depending on the probability law for $\Xi_{\mathbf{a}}$, each transition has associated with it a conditional probability $P_{\mathbf{a}}(\mathbf{x} \mid \mathbf{u}_i)$ or $P_{\mathbf{a}}(\mathbf{x} \mid \mathbf{v}_j)$. These probabilities determine the value of the optimal filter when $\mathbf{x}$ is observed. Specifically, $P_{\mathbf{a}}(\mathbf{x})$ and $P_{\mathbf{a}}(Y = 1 \mid \mathbf{x})$ can be found from $P_{\mathbf{a}}(\mathbf{x} \mid \mathbf{u}_i)$, $P_{\mathbf{a}}(\mathbf{x} \mid \mathbf{v}_j)$, $P_{\mathbf{a}}(\mathbf{u}_i)$ and $P_{\mathbf{a}}(\mathbf{v}_j)$:

$$P_{\mathbf{a}}(\mathbf{x}) = \sum_{i=1}^{m(\mathbf{x})} P_{\mathbf{a}}(\mathbf{u}_i) P_{\mathbf{a}}(\mathbf{x} \mid \mathbf{u}_i) + \sum_{j=1}^{n(\mathbf{x})} P_{\mathbf{a}}(\mathbf{v}_j) P_{\mathbf{a}}(\mathbf{x} \mid \mathbf{v}_j) \tag{1–107}$$

$$P_{\mathbf{a}}(Y = 1 \mid \mathbf{x}) = \frac{P_{\mathbf{a}}(Y = 1, \mathbf{x})}{P_{\mathbf{a}}(\mathbf{x})} = \frac{\sum_{i=1}^{m(\mathbf{x})} P_{\mathbf{a}}(\mathbf{u}_i) P_{\mathbf{a}}(\mathbf{x} \mid \mathbf{u}_i)}{\sum_{i=1}^{m(\mathbf{x})} P_{\mathbf{a}}(\mathbf{u}_i) P_{\mathbf{a}}(\mathbf{x} \mid \mathbf{u}_i) + \sum_{j=1}^{n(\mathbf{x})} P_{\mathbf{a}}(\mathbf{v}_j) P_{\mathbf{a}}(\mathbf{x} \mid \mathbf{v}_j)} \tag{1–108}$$

thereby providing an analytic formulation for the optimal filter. According to Eq. 1–105,

$$\kappa_{\mathbf{x}}(\mathbf{b}; \mathbf{a}) = \left| \sum_{i=1}^{m(\mathbf{x})} P_{\mathbf{a}}(\mathbf{u}_i) P_{\mathbf{a}}(\mathbf{x} \mid \mathbf{u}_i) - \sum_{j=1}^{n(\mathbf{x})} P_{\mathbf{a}}(\mathbf{v}_j) P_{\mathbf{a}}(\mathbf{x} \mid \mathbf{v}_j) \right| \zeta_{\mathbf{x}}(\mathbf{b}; \mathbf{a}) \tag{1–109}$$

Tractable formulations are achievable with a sparse-noise constraint [33]. *Sparse noise* relative to W is degradation for which, if $\mathbf{u}$ and $\mathbf{x}$ are the ideal and observed vectors in W_z, then $\mathbf{u}$ and $\mathbf{x}$ differ at most in a single component. At most one component of $\mathbf{u}$ is switched by $\Xi_{\mathbf{t}}$. Now denote a vector by $\mathbf{x} = (x_0, x_1, x_2, \ldots, x_{n-1})$, where x_0 is the value at z and the others are observed by raster scanning the remainder of W_z. For sparse noise, let $\mathbf{x}_k = (k, x_1, x_2, \ldots, x_{n-1})$, $k = 0, 1$, and $\mathbf{x}_{k,i}$ be the same as $\mathbf{x}_k$ except that the ith component is switched. Then $\mathcal{U}_{\mathbf{a},\mathbf{x}_0} = \{\mathbf{x}_1\}$, $\mathcal{V}_{\mathbf{a},\mathbf{x}_1} = \{\mathbf{x}_0\}$,

$$\begin{aligned} \mathcal{U}_{\mathbf{a},\mathbf{x}_1} &= \{\mathbf{x}_1, \mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \ldots, \mathbf{x}_{1,n-1}\} \\ \mathcal{V}_{\mathbf{a},\mathbf{x}_0} &= \{\mathbf{x}_0, \mathbf{x}_{0,1}, \mathbf{x}_{0,2}, \ldots, \mathbf{x}_{0,n-1}\} \end{aligned} \tag{1–110}$$

From Eq. 1–108 (and suppressing the subscript "a" in "$P_{\mathbf{a}}$" to ease notation),

$$P(Y = 1 \mid \mathbf{x}_1) = \frac{P(\mathbf{x}_1) P(\mathbf{x}_1 \mid \mathbf{x}_1) + \sum_{i=1}^{n-1} P(\mathbf{x}_{1,i}) P(\mathbf{x}_1 \mid \mathbf{x}_{1,i})}{P(\mathbf{x}_1) P(\mathbf{x}_1 \mid \mathbf{x}_1) + \sum_{i=1}^{n-1} P(\mathbf{x}_{1,i}) P(\mathbf{x}_1 \mid \mathbf{x}_{1,i}) + P(\mathbf{x}_0) P(\mathbf{x}_1 \mid \mathbf{x}_0)} \tag{1–111}$$

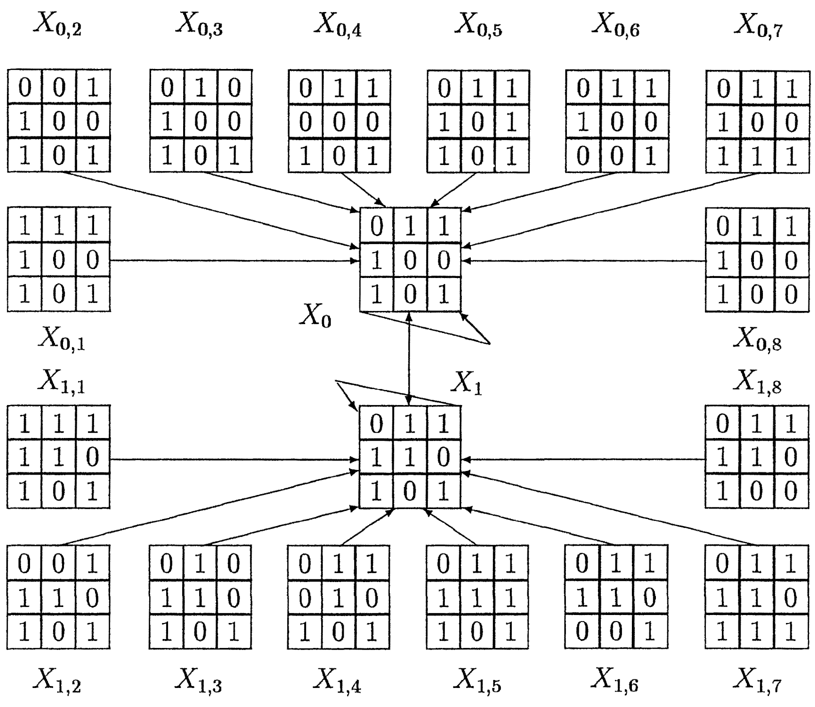


Figure 1–19. Possible transitions for sparse noise.

$$P(Y=1 \mid \mathbf{x}_0) = \frac{P(\mathbf{x}_1)P(\mathbf{x}_0 \mid \mathbf{x}_1)}{P(\mathbf{x}_1)P(\mathbf{x}_0 \mid \mathbf{x}_1) + P(\mathbf{x}_0)P(\mathbf{x}_0 \mid \mathbf{x}_0) + \sum_{i=1}^{n-1} P(\mathbf{x}_{0,i})P(\mathbf{x}_0 \mid \mathbf{x}_{0,i})} \tag{1–112}$$

thereby providing an analytic formulation of the optimal filter. Letting $\mathbf{x}^i$ denote the vector differing from $\mathbf{x}$ in component i, regardless of the center value, from Eq. 1–109,

$$\kappa_{\mathbf{x}}(\mathbf{b};\mathbf{a}) = \left| P(\mathbf{x})P(\mathbf{x} \mid \mathbf{x}) - P(\mathbf{x}^0)P(\mathbf{x} \mid \mathbf{x}^0) + \sum_{i=1}^{n-1} P(\mathbf{x}^i)P(\mathbf{x} \mid \mathbf{x}^i) \right| \zeta_{\mathbf{x}}(\mathbf{b};\mathbf{a}) \tag{1–113}$$

Figure 1–19 depicts the possible transitions for sparse noise when using a 3×3 window.

Now suppose the degradation operator is independent of the ideal image. Suppose $\mathbf{u}$ and $\mathbf{x}$ are identical except for component j, for which $u_j \neq x_j$, $0 \leqslant j \leqslant n-1$, and let γ denote complementation, so that $\gamma(x_j) = u_j$. The probability of the

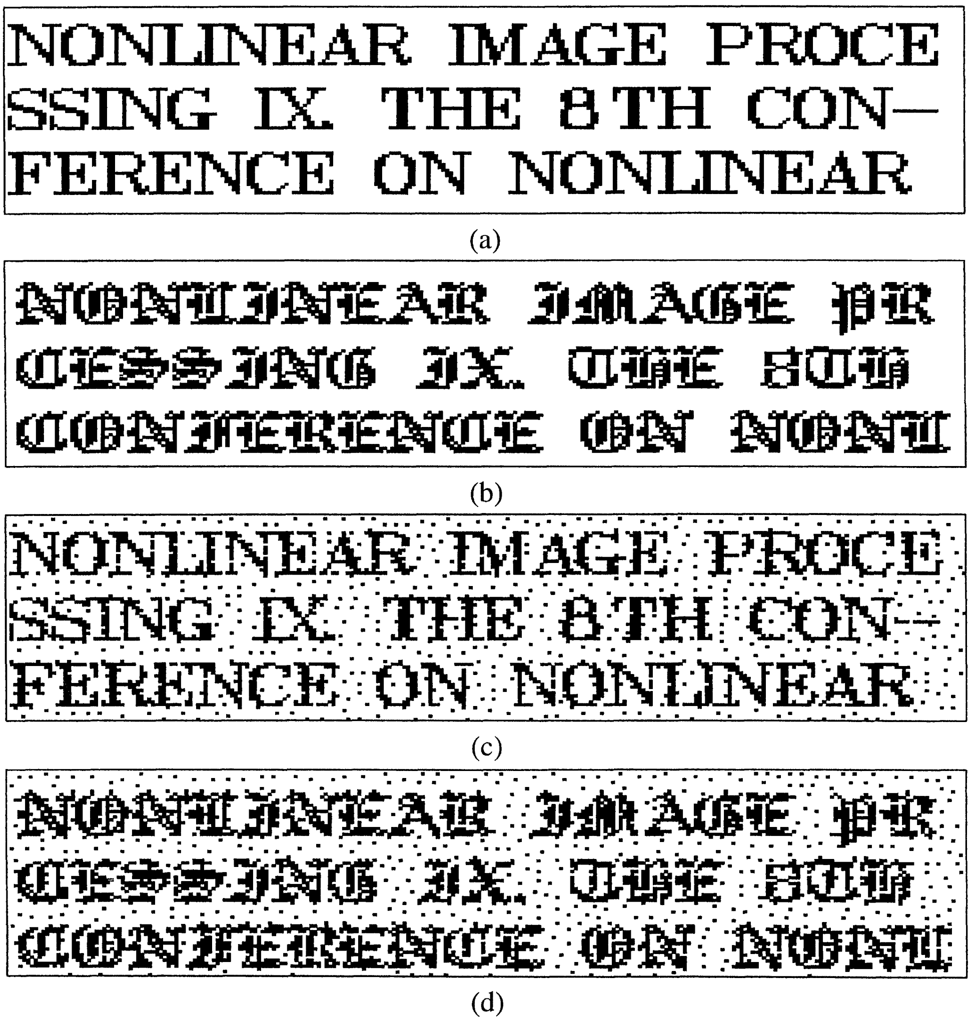


Figure 1–20. Fonts: (a) triplex; (b) gothic; (c) noisy triplex, $p = 0.053$; (d) noisy gothic, $p = 0.053$.

transition $\mathbf{u} \to \mathbf{x}$ is

$$\begin{aligned} P(\mathbf{u} \to \mathbf{x}) &= P\big(\gamma(x_j) = u_j,\ x_i = u_i \text{ for } i \neq j\big) \\ &= P\big(\gamma(x_j) = u_j\big) P\big(x_i = u_i \text{ for } i \neq j \mid \gamma(x_j) = u_j\big) \\ &= P\big(\gamma(x_j) = u_j\big) \end{aligned} \qquad (1\text{–}114)$$

where the conditional probability in the second equality is 1 owing to noise sparseness. $P(\gamma(x_j) = u_j)$ is the probability that Ξ_a flips the jth component of $\mathbf{u}$, but this is independent of $\mathbf{u}$. Thus, it is the probability that Ξ_a flips a pixel value. By stationarity, this probability, p, called the *intensity* of the independent sparse noise,

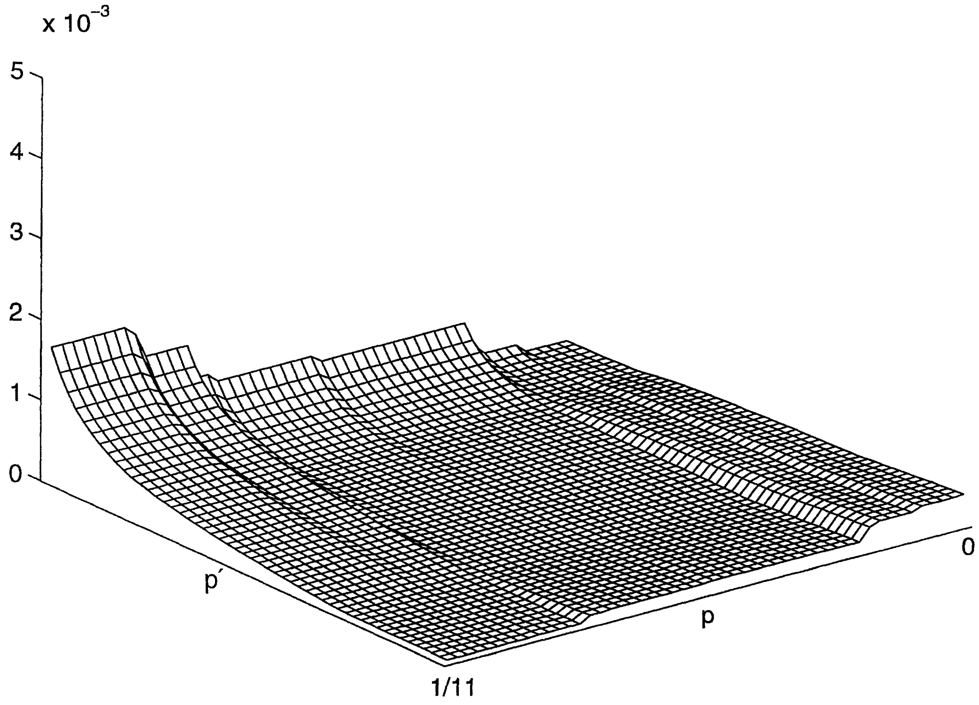


Figure 1–21. Robustness surface for triplex font.

is common across the entire domain. ($\Xi_{\mathbf{a}}$ does not necessarily act independently at each pixel, but the probability of flipping a value is common.) Consequently, for independent sparse noise, $P(\mathbf{u} \to \mathbf{x}) = p$ if $\mathbf{u} \neq \mathbf{x}$ and $P(\mathbf{u} \to \mathbf{u}) = 1 - np$. Note that $0 \leqslant p \leqslant 1/n$. Hence, $P(\mathbf{x}_k \mid \mathbf{x}_{k,i}) = p$ for $k = 0, 1$ and $i = 0, 1, \ldots, n-1$, and $P(\mathbf{x}_k \mid \mathbf{x}_k) = 1 - np$ for $k = 0, 1$. Define the ideal-image parameter

$$\lambda_{\mathbf{x}} = \sum_{i=1}^{n-1} P(\mathbf{x}^i) \tag{1–115}$$

To specify the window center in the parameter, $\lambda_{\mathbf{x},1}$ and $\lambda_{\mathbf{x},0}$ denote that $x_0 = 1$ and $x_0 = 0$, respectively. Using this parameter, Eqs. 1–111, 1–112, and 1–113 reduce to

$$P(Y = 1 \mid \mathbf{x}_1) = \frac{(1 - np)P(\mathbf{x}_1) + p\lambda_{\mathbf{x},1}}{(1 - np)P(\mathbf{x}_1) + p\lambda_{\mathbf{x},1} + pP(\mathbf{x}_0)} \tag{1–116}$$

$$P(Y = 1 \mid \mathbf{x}_0) = \frac{pP(\mathbf{x}_1)}{pP(\mathbf{x}_1) + (1 - np)P(\mathbf{x}_0) + p\lambda_{\mathbf{x},0}} \tag{1–117}$$

$$\kappa_{\mathbf{x}}(\mathbf{b}; \mathbf{a}) = \left|(1 - np)P(\mathbf{x}) + p\lambda_{\mathbf{x}} - pP\left(\mathbf{x}^0\right)\right|\zeta_{\mathbf{x}}(\mathbf{b}; \mathbf{a}) \tag{1–118}$$

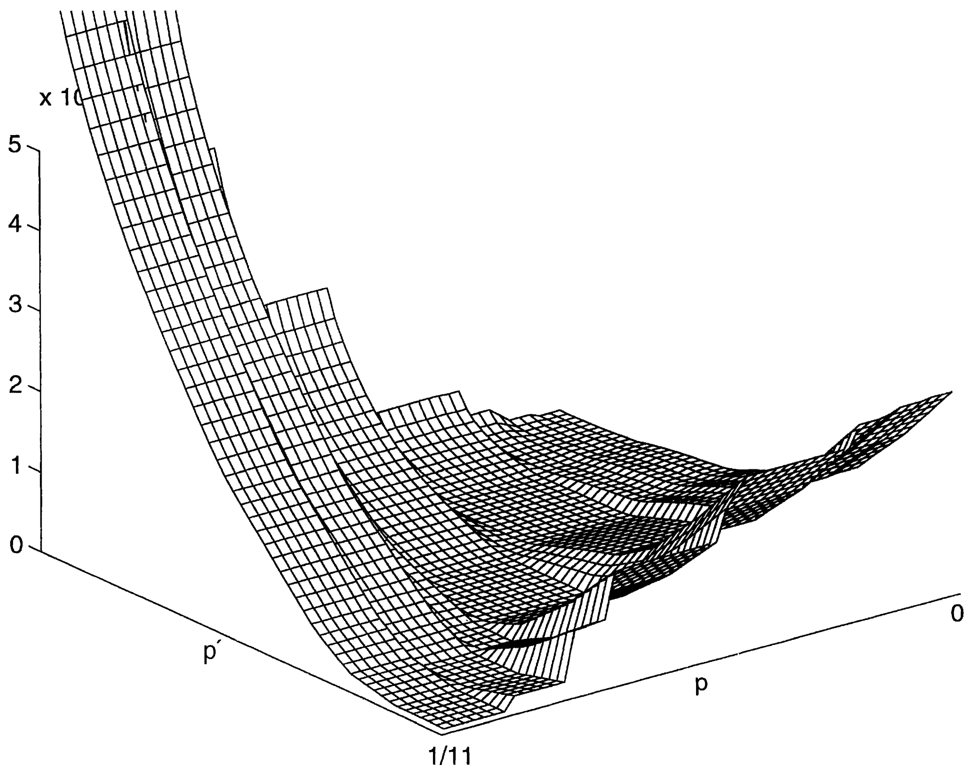


Figure 1–22. Robustness surface for gothic font.

The probabilities and robustness are analytically expressed via ideal-image parameters and p, without explicit reference to the observed process.

For independent sparse noise, robustness $\kappa(p'; p)$ is a function of two scalar variables and is geometrically represented by a *robustness surface* that is zero on the diagonal $p = p'$. For fixed p', $\kappa(p'; p)$ gives the robustness relative to design at p'. $\Psi_{p'}$ is qualitatively robust if the curve of $\kappa(p'; p)$ as a function of p is fairly flat. In terms of two variables, qualitative robustness relates to the flatness of the surface $\kappa(p'; p)$ about the diagonal. Since $\kappa(p'; p)$ only depends on ideal-image parameters and intensity, the actual degradation operator is inconsequential; only the noise intensity matters. There are various constrained random point processes that produce independent sparse noise.

EXAMPLE 1–10. We consider restoration of images degraded by independent sparse noise using a 3×3 window. Robustness for independent sparse noise is essentially analytic because it is computed from Eq. 1–118, in which only 512 probabilities $P(\mathbf{x})$ must be estimated and these can be estimated with great precision. Parts a and b of Fig. 1–20 show realizations of triplex and gothic fonts;

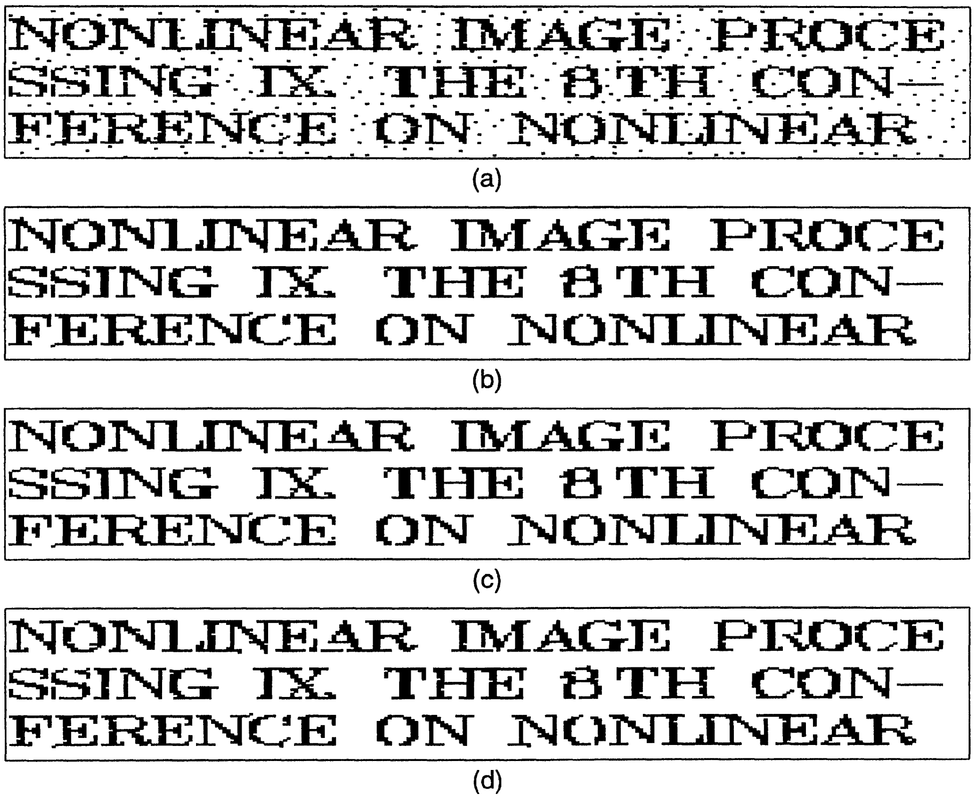


Figure 1–23. Restoration of noisy triplex font: (a) noisy triplex font, $p = 0.030$; (b) restoration by $\Psi_{0.03}$; (c) restoration by $\Psi_{0.06}$; restoration by $\Psi_{0.01}$.

parts c and d show sparse-noise-degraded versions of these realizations for intensity $p = 0.053$. Robustness surfaces for triplex and gothic fonts are shown in Figs. 1–21 and 1–22, respectively, where p values run from 1/11 out to 0 and the p' (design) axis is to the left. Figure 1–23 shows a noisy realization of triplex font for $p = 0.03$ and the results of filtering the realization by the optimal filters for $p' = p$, $p' = 0.06$, and $p' = 0.01$. Input MAE (part a) is 0.030 and MAEs for the filtered images (computed over large realizations) are 0.0123, 0.0129, and 0.0133 for $p' = p$, $p' = 0.06$, and $p' = 0.01$, respectively. Robustness values are

$$\kappa(0.03, 0.06) = 0.0006 \quad \text{and} \quad \kappa(0.03, 0.01) = 0.0010.$$

Figure 1–24 shows a noisy realization of gothic font for $p = 0.03$ and the results of filtering the realization by the optimal filters for $p' = p$, $p' = 0.06$, and $p' = 0.01$. Input MAE is 0.030 and MAEs for the filtered images are 0.0167, 0.0182, and 0.0189 for $p' = p$, $p' = 0.06$, and $p' = 0.01$, respectively,

$$\kappa(0.03, 0.06) = 0.0015 \quad \text{and} \quad \kappa(0.03, 0.01) = 0.0022.$$

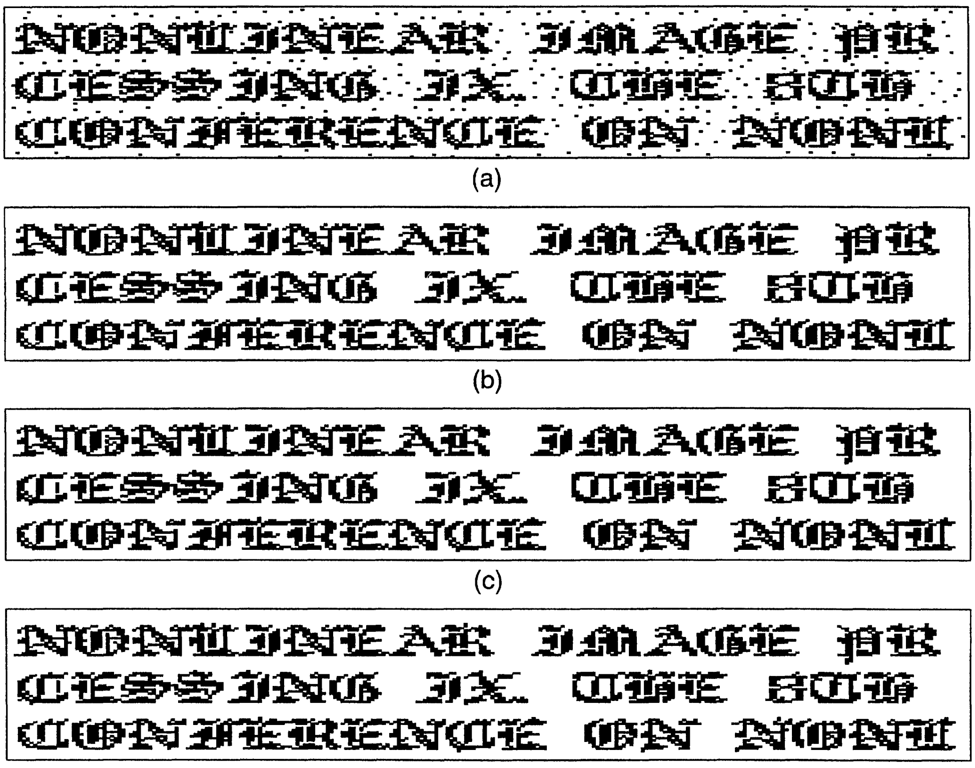


Figure 1–24. Restoration of noisy gothic font: (a) noisy triplex font, $p = 0.030$; (b) restoration by $\Psi_{0.03}$; (c) restoration by $\Psi_{0.06}$; restoration by $\Psi_{0.01}$.

REFERENCES

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

[2] Maragos, P., and R. Schafer, "Morphological Filters – Part I: Their Set-Theoretic Analysis and Relations to Linear Shift-Invariant Filters," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, **35**, 1987.

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

[4] Dougherty, E. R., and R. M. Haralick, "Unification of Nonlinear Filtering in the Context of Binary Logical Calculus — Part I: Binary Filters," *Mathematical Imaging and Vision*, **2** (2), 1992.

[5] Serra, J., *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.

[6] Banon, G. J. F., and J. Barrera, "Minimal Representation for Translation Invariant Set Mappings by Mathematical Morphology," *SIAM J. Applied Mathematics*, **51**, 1991.

[7] Goutsias, J., "Morphological Analysis of Discrete Random Shapes," *Mathematical Imaging and Vision*, **2** (2/3), 1992.

[8] Barrera, J., Dougherty, E. R., and N. S. Tomita, "Automatic Programming of Binary Morphological Machines by Design of Statistically Optimal Operators in the Context of Computational Learning Theory," *Electronic Imaging*, **6** (1), 1997.

[9] Dougherty, E. R., and R. P. Loce, "Optimal Binary Differencing Filters: Design, Logic Complexity, Precision Analysis, and Application to Digital Document Processing," *Electronic Imaging*, **5** (1), 1996.

[10] Loce, R. P., and E. R. Dougherty, *Enhancement and Restoration of Digital Documents: Statistical Design of Nonlinear Algorithms*, SPIE Press, Bellingham, 1997.

[11] Dougherty, E. R., and R. P. Loce, "Optimal Mean-Absolute-Error Hit-or-Miss Filters: Morphological Representation and Estimation of the Binary Conditional Expectation," *Optical Engineering*, **32** (4), 1993.

[12] Dougherty, E. R., and R. P. Loce, "Precision of Morphological-Representation Estimators for Translation-Invariant Binary Filters: Increasing and Nonincreasing," *Signal Processing*, **40** (3), 1994.

[13] McCluskey, E. J., " Minimization of Boolean Functions," *Bell System Tech. J.*, **35** (5), 1956.

[14] Quine, W. V., "The Problem of Simplifying Truth Functions," *American Math. Monthly*, **59** (8), 1952.

[15] Barrera, J., Terada, R., Côrrea da Silva, F. S., and N. S. Tomita, "Automatic Programming of Morphological Machines for OCR," *Mathematical Morphology and its Applications to Image and Signal Processing*, Atlanta, 1996.

[16] Schmitt, M., *Des algorithmes morphologiques a l'a intelligence artificielle*, These present a l'Ecole Superieur des Mines de Paris, Fontainebleau, 1989.

[17] Dougherty, E. R., "Optimal Mean-Square *N*-Observation Digital Morphological Filters – Part I: Optimal Binary Filters," *CVGIP: Image Understanding*, **55** (1), 1992.

[18] Loce, R. P., and E. R. Dougherty, "Facilitation of Optimal Binary Morphological Filter Design Via Structuring-Element Libraries and Observation Constraints," *Optical Engineering*, **31** (5), 1992.

[19] Loce, R. P., and E. R. Dougherty, "Optimal Morphological Restoration: The Morphological Filter Mean-Absolute-Error Theorem," *Visual Communication and Image Representation*, **3** (4), 1992.

[20] Dougherty, E. R., and R. P. Loce, "Efficient Design Strategies for the Optimal Binary Digital Morphological Filter: Probabilities, Constraints, and Structuring-Element Libraries," in *Mathematical Morphology in Image Processing*, ed. E. R. Dougherty, Marcel Dekker, New York, 1993.

[21] Mathew, A. V., Dougherty, E. R., and V. Swarnakar, "Efficient Derivation of the Optimal Mean-Square Binary Morphological Filter from the Conditional Expectation via a Switching Algorithm for the Discrete Power-Set Lattice," *Circuits, Systems, and Signal Processing*, **12** (3) 1993.

[22] Han, C. C., and K. C. Fan, "A Greedy and Branch & Bound Searching Algorithm for Finding the Optimal Morphological Filter on Binary Images," *IEEE Signal Processing Letters*, **1**, 1994.

[23] Dougherty, E. R., Zhang, Y., and Y. Chen, "Optimal Iterative Increasing Binary Morphological Filters," *Optical Engineering*, **35** (12), 1996.

[24] Zhang, Y., Loce, R. P., and E. R. Dougherty, "Document Enhancememt Using Optimal Iterative and Paired Morphological Filters," *Proc. SPIE*, **3027**, 1997.

[25] Haussler, D., "Decision Theoretic Generalizations of the PAC Model for Neural Nets and Other Learning Applications," *Information and Computation*, **100**, 1992.

[26] Barrera, J., Tomita, N. S., and F. S. Côrrea da Silva, "Automatic Programming of Morphological Machines by PAC Learning," *Proc. SPIE*, **2568**, 1995.

[27] Valiant, L., "A Theory of the Learnable," *Comm. ACM*, **27**, 1984.

[28] Kong, T. Y., and A. Rosenfeld, "Digital Topology: Introduction and Survey," *CVGIP*, **48**, 1989.

[29] Crimmons, T., and W. Brown, "Image Algebra and Automatic Shape Recognition," *IEEE Trans. Aerospace and Electronic Systems*, **21**, 1985.

[30] Zhao, D., and D. Daut, "Morphological Hit-or-Miss Transformation for Shape Recognition," *Visual Communication and Image Representation*, **2** (3), 1991.

[31] Dougherty, E. R., and D. Zhao, "Model-Based Characterization of Statistically Optimal Design for Morphological Shape Recognition Algorithms via the Hit-or-Miss Transform," *Visual Communication and Image Representation*, **3** (2), 1992.

[32] Barrera, J., Dougherty, E. R., and N. S. Hirata, "Design of Optimal Morphological Operators Using Prior Filters," *Acta Stereologica*, **16** (3), 1998.

[33] Grigoryan, A. M., and E. R. Dougherty, "Robustness of Optimal Binary Filters," *Electronic Imaging*, **7** (1), January, 1998.