DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Relatório Técnico

RT-MAC-9603

Automatic Programming of Binary Morphological Machines by Design of Statistically Optimal Operators in the Context of Computational Learning Theory

Junior Barrera
Edward R. Dougherty
Nina Sumiko Tomita

Abril 96

Automatic Programming of Binary Morphological Machines by Design of Statistically Optimal Operators in the Context of Computational Learning Theory

Junior Barrera[1], Edward R. Dougherty[2], Nina Sumiko Tomita[3,4]

ABSTRACT

Representation of set operators by artificial neural networks and design of such operators by inference of network parameters is a popular technique in binary image analysis. In this paper, we propose an alternative to this technique: automatic programming of morphological machines (MMach's) by the design of statistically optimal operators. We propose a formulation of the procedure for designing set operators that extends the one stated by Dougherty for binary image restoration, show the relation of this new formulation with the one stated by Haussler for learning Boolean concepts in the context of machine learning theory (that usually is applied to neural networks), present a new learning algorithm for Boolean concepts represented as MMach programs, and give some application examples in binary image analysis.

[1] Departmento de Ciencia da Computacao, Universidade de Sao Paulo, Sao Paulo, Brazil
[2] Center for Imaging Science, Rochester Institute of Technology, Rochester, NY.
[3] Departmento de Ciencia da Computacao, Universidade de Sao Paulo, Sao Paulo, Brazil

[4] This work was partialy supported by Olivetti do Brasil, PROTEM-CC-ANIMOMAT e Cooperation USP-COFECUB.

1. Introduction

Binary image analysis is an important tool for various imaging applications, including industrial process control, office automation, and quantitative microscopy. A central problem in binary image analysis is the design of efficient image processing procedures to perform desired tasks. Historically, design of imaging procedures has been performed on an ad hoc basis with algorithm performance depending on a designer's knowledge of and experience with image processing tools. *Ad hoc* design restricts the users of image analysis systems to experts in image processing — or, worse, to experts in some particular branch of image processing. To overcome this limitation, recent research has addressed the problem of automatic generation of image processing procedures. The goal is to find suitable knowledge representation formalisms and develop tools that translate them into efficient image processing procedures. Some of these tools use examples (collections of input-output pairs of images) as the knowledge source for representation formalism.

A natural model for a procedure used in binary image analysis is a set mapping (operator) applied to a discrete random set (Goutsias [16]). In this framework, procedure design from examples can be modeled by statistical estimation of set operators from observations of input-output image pairs. Formulation of this statistical approach can be decomposed into two basic steps: (1) estimation of the operator input **S** and ideal output **I** random sets; (2) design of a set operator Ψ such that $\Psi(\mathbf{S})$ is statistically close to **I**.

The second step can be formulated as an optimization problem: given a family $\mathfrak{I}$ of set operators, where the closeness of the ideal **I** and estimated $\Psi(\mathbf{S})$ random sets is measured by some probabilistic error measure, an element $\Psi_{opt} \in \mathfrak{I}$ is called *optimal* in $\mathfrak{I}$ if it posseses minimum error. If all elements of $\mathfrak{I}$ can be characterized by some algebraic representation or formal language, then optimization can be viewed as finding an efficient representation defining an optimal operator.

Artificial neural networks (Hassoun [18]) provide a general framework to study set operators. An artificial neural net is a parallel computational model comprised of densely interconnected adaptive processing units, called *neurons*. A neuron is a multiple input and multiple output device. It receives excitatory and inhibitory inputs from other neurons, integrates these inputs and, when the result of the integration exceeds a given threshold, sends excitatory or inhibitory inputs to other neurons. There are many possible architectures for neural nets. In the feedforward architecture, the neural net is organized as layers of neurons that receive inputs from another layer, process the information and send the outputs to different layers. This is a general architecture that is sufficient to represent any set operator. Using neural nets to represent set operators, the optimization procedure consists of finding a net architecture and estimating the parameters of the neurons of this net that characterize an optimal operator.

Mathematical morphology (Serra [31]) is another general framework to study set operators. Two simple families of operators are *erosions* and *dilations*, and these are characterized by subsets called *structuring elements*. A central paradigm in mathematical morphology is representation of set operators by concatenations of erosions and dilations via the operations of composition, intersection, union, and complementation. This paradigm can be stated by the use of a formal language, called the *morphological language* (Barrera [5]), whose vocabulary is composed of erosions, dilations, intersection, union and complementation. This language is *complete* (i.e., it is enough to describe any

set operator) and *expressive* (i.e. most useful operators can be described by relatively few words). A phrase in the morphological language is called a *morphological operator*. Since the 1960s, special machines, called *morphological machines* [*MMach's*] (Klein [20, 21], Barrera [6]), have been built to implement this language. A MMach program is just an implementation of a morphological operator. Using the morphological language to represent set operators, the optimization procedure consists of finding a phrase structure and estimating the structuring elements that characterize an optimal operator. In this context, the complete design procedure (steps 1 and 2 above) is called *automatic programming of morphological machines* (Barrera [7]).

An important family of operators is the family of *W-operators*. These are translation invariant and locally defined operators. Any *W*-operator can be represented by a single phrase structure, called the *standard morphological representation* (Banon [3], Barrera [9]) and the argument of the operator at a particular pixel consists of random pixel values in a predetermined window *W* translated to the given pixel. Restricted to *W*-operators in the standard morphological representation, optimization can be seen as estimation of the structuring elements of this representation.

In this paper, we present a formulation of the procedure for designing set operators that extends the ones stated by Dougherty [10] and Dougherty and Loce [11] for the respective cases of increasing and nonincreasing optimal binary set operators image restoration; we discuss the relation between the proposed formulation and the one stated by Haussler [19] for learning Boolean concepts in the context of machine learning theory (a formulation often applied to neural nets); and we provide a new learning algorithm for Boolean concepts (or, equivalently, *W*-operators) represented as MMach programs.

Following this introduction, Sections 2 presents the proposed statistical model. Section 3 recalls the standard morphological representation of W-operators. Section 4 specializes the proposed statistical model for the family of W-operators. Section 5 discusses suboptimality. Section 6 gives analytical expressions for the precision of estimation of W-operators. Section 7 shows the relation between machine learning theory and the proposed formulation. Section 8 discusses the modeling of prior information. Section 9 presents the proposed algorithm for learning Boolean concepts. Section 10 gives some application examples in binary image analysis. Finally, we discuss some aspects of the results presented and propose some future steps for this research.

2. System Model

The central thrust of the current paper is to explain the manner in which set operators are automatically and optimally designed to estimate an image when it is observed after going through some system. Images are modeled as discrete random processes (sets). If Ψ is a binary image (set) operator, then for each random image **S** there is an output random image $\Psi(\mathbf{S})$. If S is any observed realization of **S**, then $\Psi(S)$ is a realization of $\Psi(\mathbf{S})$. The task is to design an operator Ψ so that, given the input process **S**, $\Psi(\mathbf{S})$ is probabilistically close to some desired process **I**. We shall call **S**, **I**, and $\Psi(\mathbf{S})$ the *observation*, *ideal*, and *estimator* processes, respectively.

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

$$\varepsilon[\mathbf{I}, \Psi_{opt}(\mathbf{S})] \leq \varepsilon[\mathbf{I}, \Psi(\mathbf{S})] \quad (1)$$

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

Operationally, 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. Optimization involves minimizing the error $\varepsilon[\mathbf{I}, \Psi(\Xi(\mathbf{I}))]$. In general, Ξ is a multivalued image operator, meaning that, given a realization I of the ideal image, there are many possible output realizations $\Xi(I)$.

Set-theoretic system models. A much-studied application of the method is when **I** represents some ideal image process and Ξ is a degradation (noise) transformation — the ideal image is obscured by noise [11,12, 23-25]. Ξ can take many forms, any particular form depending on the physical system causing the degradation. For instance, if **I** is an ideal binary image process and **N** is some other binary process, then the *signal-union-noise model* is defined by the degradation transformation

$$\mathbf{S} = \Xi_{\cup}(\mathbf{I}) = \mathbf{I} \cup \mathbf{N} \quad (2)$$

Each observed realization is of the form $S = I \cup N$, where I and N are realizations of **I** and **N**, respectively. Unless there are no restrictions placed on the ideal and noise processes, Eq. 2 does not fully define the system transformation. For instance, it might be required that **I** and **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 process involving both the adjoining and deletion of pixels, the system transformation being

$$\Xi_{\cup,-}(\mathbf{I}) = (\mathbf{I} \cup \mathbf{N}_1) - \mathbf{N}_2 \quad (3)$$

Morphological system models. Another system model is the *dilation/erosion model*. If we assume a structuring element B contains the origin, then, for any realization I of the ideal image process, $I \oplus B \supset I$ and $I \ominus B \subset I$, where $\oplus$ and $\ominus$ denote dilation and erosion, respectively. Since these inclusion relations hold for any realizations, the observation process satisfies the relations $\mathbf{I} \oplus B \supset \mathbf{I}$ and $\mathbf{I} \ominus B \subset \mathbf{I}$. Whether or not $0 \in B$, if we define the system transformation by $\Xi_{\oplus}(\mathbf{I}) = \mathbf{I} \oplus B$ or by $\Xi_{\ominus}(\mathbf{I}) = \mathbf{I} \ominus B$, 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 **B**, in which case the dilation and erosion system transformations take the forms $\Xi_{\oplus}(\mathbf{I}) = \mathbf{I} \oplus \mathbf{B}$ and $\Xi_{\ominus}(\mathbf{I}) =$

$\mathbf{I} \ominus \mathbf{B}$, respectively, both which multivalued. The dilation/erosion model is not logically distinct from the union/subtraction model since, if $0 \in B$, then

$$\mathbf{I} \oplus \mathbf{B} = \mathbf{I} \cup [(\mathbf{I} \oplus \mathbf{B}) - \mathbf{I}] \tag{4}$$

so that the dilation model is a union model with noise process $\mathbf{N} = (\mathbf{I} \oplus \mathbf{B}) - \mathbf{I}$. A dual statement applies to the erosion model, with erosion being expressed as a subtraction model when $0 \in \mathbf{B}$.

System models can be defined by other morphological operations. For instance, if $\mathbf{B}_1$ and $\mathbf{B}_2$ are random structuring elements, then the system transformation might be erosion followed by dilation or dilation followed by erosion,

$$\Xi_{\ominus,\oplus}(\mathbf{I}) = (\mathbf{I} \ominus \mathbf{B}_1) \oplus \mathbf{B}_2 \tag{5}$$

$$\Xi_{\oplus,\ominus}(\mathbf{I}) = (\mathbf{I} \oplus \mathbf{B}_1) \ominus \mathbf{B}_2 \tag{6}$$

respectively. If $\mathbf{B}_1 = \mathbf{B}_2 = \mathbf{B}$, then these system transformations are, respectively, *opening* and *closing* by $\mathbf{B}$, $\Xi_\circ(\mathbf{I}) = \mathbf{I} \circ \mathbf{B}$ and $\Xi_\bullet(\mathbf{I}) = \mathbf{I} \bullet \mathbf{B}$.

Edge detection. To characterize binary edge detection in terms of a system model, we assume the existence of a canonical edge detector: given a binary deterministic image L, there is an edge operator Γ such that $\Gamma(L)$ 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 L_I having edge I [$\Gamma(L_I) = I$] and we know the algorithm H that produces L_I from I. For instance, Γ might be the 3×3 dilation of the image minus the image and H might be a contour filler. If the system function is merely equal to H, then perfect edge construction would result from applying Γ and there would be no operator-design problem. A more practical model results by assuming the system model is H followed by noise degradation. If the noise is union noise, then the system transformation takes the form $\Xi_{edge}(\mathbf{I}) = \mathrm{H}(\mathbf{I}) \cup \mathbf{N}$. An optimal edge detector minimizes the error

$$\varepsilon[\mathbf{I}, \Psi(\Xi_{edge}(\mathbf{I}))] = \varepsilon[\mathbf{I}, \Psi(\mathrm{H}(\mathbf{I}) \cup \mathbf{N})] \tag{7}$$

Matched filtering. 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 of translated realizations $A_1, A_2, \ldots, A_n$ of $\mathbf{A}$ to the points $z_1, z_2, \ldots, z_n$, respectively. As a random process, the observation image is defined by the system transformation

$$\mathbf{S} = \Xi_{match}(\mathbf{I}) = \bigcup_{i=1}^{N} \mathbf{A}_i + \mathbf{z}_i \tag{8}$$

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 overlap (intersect) or, if they do overlap, overlapping 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: there exists a deterministic set A_0 such that $\mathbf{A} = A_0 \cup \mathbf{N}$. It might also 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. 8 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

$$\Xi_{match,m}(\mathbf{I}) = \left(\bigcup_{i=1}^{N} \mathbf{A}_i + \mathbf{z}_i\right) \cup \left(\bigcup_{k=1}^{m} \bigcup_{j=1}^{N(k)} \mathbf{A}_j^k + \mathbf{z}_{k,j}\right) \tag{9}$$

where $N_{(1)}, N_{(2)}, \ldots, N_{(m)}$ are the random 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, for $j = 1, 2, \ldots, N(k)$, $\mathbf{A}_j^k$ is identically distributed to $\mathbf{A}^k$.

Algorithm emulation. Suppose we wish to construct a window-based operator to emulate an algorithm, say A; that is, we desire a window operator that optimally estimates 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 Ψ of algorithm A minimizes the error of emulation,

$$\varepsilon[\mathbf{I}, \Psi(\Xi_{\mathrm{A}}(\mathbf{I}))] = \varepsilon[\mathbf{I}, \Psi(\mathrm{A}^{-1}(\mathbf{I}))] \tag{10}$$

If A is a segmentation algorithm and the emulator estimates the segmentation, then $\mathbf{I}$ is composed of segmented images from $\mathbf{R}$ and $\mathrm{A}^{-1}(\mathbf{I})$ consists of unsegmented images.

One situation that can be treated by algorithm emulation is adaptation of an algorithm to a noisy environment. Suppose $\mathrm{A}^{-1}(\mathbf{I})$ is the observation process resulting directly from algorithm inversion. As designed originally, A is meant to be applied to this process. But suppose the observations have been degraded. Then the system transformation is given by $\Xi_{\mathrm{A}}(\mathbf{I}) = \Phi(\mathrm{A}^{-1}(\mathbf{I}))$, where Φ is a noise-degradation operator. For instance, if the presegmented images are degraded by dilation by a random structuring element $\mathbf{B}$, then $\Xi_{\mathrm{A}}(\mathbf{I}) = \mathrm{A}^{-1}(\mathbf{I}) \oplus \mathbf{B}$ and the optimal emulator is the filter Ψ minimizing the emulation error $\varepsilon[\mathbf{I}, \Psi(\mathrm{A}^{-1}(\mathbf{I}) \oplus \mathbf{B})]$. With direct emulation using the system function Ξ_{A}, the only advantage of emulation is the translation of the algorithm into an MMach program that represents a window 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 optimal emulation will optimally estimate the algorithm as if it

were applied in a nondegraded environment. For example, this method might be used to transform a neural net into an improved MMach program.

Probability structure. Since we have assumed a finite grid, the ideal image process **I** has a finite number of realizations, with each realization I having a probability $P(I)$ of occurrence and

$$\sum_{I \in \mathbf{I}} P(I) = 1 \tag{11}$$

The system transformation is multivalued but not all observations resulting from an ideal realization need have equal probability. Moreover, a given observation realization might result from system action on a number of ideal realizations. To quantify matters, suppose **I** has n realizations $I_1, I_2, \ldots, I_n$ and, for $k = 1, 2, \ldots, n$, $\Xi(I_k)$ can be any of $n(k)$ realizations,

$$\Xi(I_k) \in \{S_{k,1}, S_{k,2}, \ldots, S_{k,n(k)}\} \tag{12}$$

If S is a realization of the observation process, then

$$S \in \bigcup_{k=1}^{n} \Xi(I_k) = \bigcup_{k=1}^{n} \{S_{k,1}, S_{k,2}, \ldots, S_{k,n(k)}\} \tag{13}$$

$\Xi(I_1), \Xi(I_2), \ldots, \Xi(I_n)$ need not form a partition of **S** because they need not be mutually disjoint — there can exist realizations $S_{k,j}$ and $S_{l,r}$, $k \neq l$, such that $S_{k,j} = S_{l,r}$. Given I_k, each realization $S_{k,j}$ has a conditional probability $P_\Xi(S_{k,j} \mid I_k)$ of resulting from Ξ.

For any observation realization S, its probability of resulting Ξ is given by

$$P_\Xi(S) = \sum_{\{k:\, S \in \Xi(I_k)\}} \sum_{\{j:\, S = S_{k,j}\}} P_\Xi(S_{k,j} | I_k) \tag{14}$$

There is one term in the double sum for each possible occurrence of S resulting from the system transformation. According to Bayes' formula, given an observation S, the probability that that the ideal process has realization I_k is given by

$$P_\Xi(I_k | S) = \frac{P_\Xi(S | I_k) P(I_k)}{\sum_{r=1}^{n} P_\Xi(S | I_r) P(I_r)} \tag{15}$$

A Bayesian approach would be to define $\Psi(S)$ to be the ideal realization I_k that maximizes $P_\Xi(I_k \mid S)$. Typically, there are too many realizations to make this approach feasible. Also, as formulated, the family $\mathfrak{I}$ of set operators consists of all set operators. To obtain a practical formulation, we will take $\mathfrak{I}$ as the family of W-operators

3. Standard Morphological Representation of W-operators

To design translation-invariant windowed digital filters we will employ pixelwise error and base design on morphological representation. The present section gives the basics of binary morphological representation, tying it directly to classical AND-OR representation of logical functions. Let $W = \{w_1, w_2,..., w_n\}$ be a finite window (set of pixels),

$$W_z = \{w_1 + z, w_2 + z,..., w_n + z\} \tag{16}$$

be the translation of W to the pixel z, and S be a digital image. The set operator Ψ is translation-invariant with window W (a W-operator), if there exists a window function ψ defined on n variables such that the filter output $\Psi(S)$ is defined at pixel z by

$$\begin{aligned}\Psi(S)(z) &= \psi(S \cap W_z) \\ &= \psi(S(w_1 + z), S(w_2 + z),..., S(w_n + z))\end{aligned} \tag{17}$$

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.

Since each variable is binary-valued, ψ is a Boolean (logical) function on n binary values and has a logical sum-of-products *disjunctive-normal-form* representation in terms of the n variables $x_1, x_2,..., x_n$ in the translated window. Letting $S(w_k + z) = x_k$,

$$\psi(x_1, x_2,..., x_n) = \sum_i x_1^{p(i,1)} x_2^{p(i,2)} \cdots x_n^{p(i,n)} \tag{18}$$

where the "sum" denotes OR, the "product" denotes AND, $p(i, k)$ is either "c" or null depending on whether the variable is complemented or not complemented, and there are at most 2^n terms in the expansion. The representation can be (nonuniquely) reduced by logic reduction to a sum of products containing a minimal number of logic gates. Then

$$\psi(x_1, x_2,..., x_n) = \sum_i x_1^{p(i,1)} x_2^{p(i,2)} \cdots x_{m(i)}^{p(i,m(i))} \tag{19}$$

The classical Boolean representations have corresponding morphological forms. Suppose Ψ is a W-operator and L and U are binary functions defined on W such that $L \le U$. The *interval* defined by the pair (L, U) is the set of all binary functions V defined on W such that $L \le V \le U$. 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 the pair (L, U) is defined at pixel z by

$$[S \otimes (L, U)](z) = \begin{cases} 1, & \text{if } L_z \le S \cap W_z \le U_z \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

where L_z and U_z denote the functions L and U operating on the translated window W_z. If $L = U$, then the pair is said to be *canonical* and $[S \otimes (U, L)](z) = [S \otimes (U, U)](z) = 1$ if and only if $S \cap W_z = U_z$, which means there is an exact pattern match.

Classically, the hit-or-miss transform has been expressed differently (but equivalently) For subsets $E, F \subset W$, the hit-or-miss transform has been defined by

$$[S \otimes (E, F)](z) = \begin{cases} 1, & \text{if } E_z \subset S \cap W_z \text{ and } F_z \subset S^c \cap W_z \\ 0, & \text{otherwise} \end{cases}$$

$$= \begin{cases} 1, & \text{if } E_z \subset S \cap W_z \text{ and } F_z \cap (S \cap W_z) = \varnothing \\ 0, & \text{otherwise} \end{cases} \qquad (21)$$

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 $L_z \le S \cap W_z \le U_z$ if and only if $E_z \subset S \cap W_z$ and $F_z \cap (S \cap W_z) = \varnothing$.

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

$$E = L = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad U = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \qquad F = U^c = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Then

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

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

$$(E, F) = \begin{pmatrix} 1 & 1 & 1 \\ \times & 1 & \times \\ 0 & 0 & 0 \end{pmatrix}$$

Corresponding to the logical sum-of-product representation (Eq. 19) is the morphological union-of-hit-or-miss representation for a W-operator (Banon [3]),

$$\Psi(S) = \bigcup_{(E,F) \in \mathbf{C}_\Psi} S \otimes (E, F) \qquad (22)$$

where C_Ψ denotes the collection of structuring pairs defining the operator Ψ. The result of Eq. 22 is called the *standard morphological representation*. The exact correspondence with Eq. 19 results from recognizing that there is a one-to-one correspondence between set pairs and the logical products:

$$(E, F) \leftrightarrow x_1^{p(i,1)} x_2^{p(i,2)} \cdots x_{n(i)}^{p(i,n(i))} \tag{23}$$

where x_k corresponds to the pixel $w_k \in W$, x_k appears uncomplemented if $w_k \in E$, x_k appears complemented if $w_k \in F$, and x_k does not appear in the product if $x_k \notin E \cup F$. It is immediate from this correspondence that a structuring pair is canonical if and only if the corresponding logical product contains all n variables in the window.

4. Optimal W-operators

For operator optimization, let l be a *loss function* defined from $\{0,1\} \times \{0,1\} \rightarrow [0, 1]$, where $l(a, b)$ measures the cost of the difference between a and b. When $\mathbf{Y}$ is a random binary process, $\mathbf{X}$ is a (not necessarily binary) random process in a given class of random processes, and h is a binary function defined on this class, we will write $l(\mathbf{Y}, h(\mathbf{X}))$. Our concern is measuring the goodness of $h(\mathbf{X})$ as an estimator of $\mathbf{Y}$. To design set operators in conjunction with morphological representation, we shall define error pixelwise by

$$\varepsilon[\mathbf{I}, \Psi(\Xi(\mathbf{I}))] = \frac{1}{M} \sum_{z \in D} E[l(\mathbf{I}(z), \Psi(\Xi(\mathbf{I}))(z))] \tag{24}$$

where D is the domain (image frame) of $\mathbf{I}$, M is the number of pixels in D, and l is a *loss function* specifying the cost of the difference between $\mathbf{I}(z)$ and $\Psi(\Xi(\mathbf{I}))(z)$.

If we make no further assumptions on $\mathbf{I}$ and $\Psi(\Xi(\mathbf{I}))$, then the optimal operator (filter) chosen from the family of translation-invariant set operators is not necessarily the same operator that would be chosen if the family considered were the family of all possible set operators (not necessarily translation invariant). However, if $\mathbf{I}$ and $\Psi(\Xi(\mathbf{I}))$ are assumed to be jointly stationary, then the error expression reduces to

$$\varepsilon[\mathbf{I}, \Psi(\Xi(\mathbf{I}))] = E[l(\mathbf{I}(z), \Psi(\Xi(\mathbf{I}))(z))] \tag{25}$$

where z is an arbitrary pixel, and the set operator chosen from the family of translation-invariant set operators is the same as the one chosen if optimization is over all possible set operators. Stationarity is a modeling assumption commonly used to make both design and digital implementation feasible (as is wide-sense stationarity in the case of optimal linear filters). When $\mathbf{I}$ and $\Psi(\Xi(\mathbf{I}))$ are jointly stationary, z (the pixel where the error is evaluated) is arbitrary and need not be specified, so that the pixel z can be omitted from Eq. 25. Since our concern is with translation-invariant operators, we assume stationarity. We shall also assume Ξ is fixed and write the observed process as $\mathbf{S}$.

The loss function l defined by

$$l(\mathbf{I}(z), \Psi(\Xi(\mathbf{I}))(z)) = |\mathbf{I}(z) - \Psi(\Xi(\mathbf{I}))(z)| \tag{26}$$

is the *mean absolute* loss function. The associated error is the *mean absolute error* (*MAE*) and is denoted by $MAE\langle\Psi\rangle$. Under stationary conditions, we write

$$MAE\langle\Psi\rangle = E[|\mathbf{I} - \Psi(\Xi(\mathbf{I}))|] \tag{27}$$

The optimal filter corresponding to minimum MAE is the *binary conditional expectation*,

$$\Psi_{MAE}(\mathbf{S})(z) = \begin{cases} 1, \text{ if } E[\mathbf{I}(z)|\mathbf{S} \cap W_z] > 0.5 \\ 0, \text{ if } E[\mathbf{I}(z)|\mathbf{S} \cap W_z] \le 0.5 \end{cases} \tag{28}$$

where $\mathbf{S} \cap W_z$ is $\mathbf{S}$ restricted to the window W translated to z. Assuming some canonical way of reading the values in the translated window, $\mathbf{S} \cap W_z$ is a binary random vector whose components are given by the window values. Because images are binary,

$$E[\mathbf{I}(z)|\mathbf{S} \cap W_z] = P(\mathbf{I}(z) = 1|\mathbf{S} \cap W_z) \tag{29}$$

and the binary conditional expectation can be rewritten as

$$\Psi_{MAE}(\mathbf{S})(z) = \begin{cases} 1, \text{ if } P(\mathbf{I}(z) = 1|\mathbf{S} \cap W_z) > 0.5 \\ 0, \text{ if } P(\mathbf{I}(z) = 1|\mathbf{S} \cap W_z) \le 0.5 \end{cases} \tag{30}$$

Denoting a deterministic realization of the binary vector $\mathbf{S} \cap W_z$ by $\mathbf{x}$, some algebra yields

$$\Psi_{MAE}(\mathbf{S})(z) =$$

$$\sum_{\{\mathbf{x}:P(I=1|\mathbf{x})>0.5\}} P(\mathbf{S} \cap W_z = \mathbf{x})P(\mathbf{I}(z) = 0|\mathbf{x}) \; + \sum_{\{\mathbf{x}:P(I=1|\mathbf{x})\le 0.5\}} P(\mathbf{S} \cap W_z = \mathbf{x})P(\mathbf{I}(z) = 1|\mathbf{x}) \tag{31}$$

If we list the possible realizations of $\mathbf{S} \cap W_z$ in a table along with the *a priori* probabilities $P(\mathbf{S} \cap W_z = \mathbf{x})$ and the conditional probabilities $P(\mathbf{I}(z) = 0|\mathbf{x})$ and $P(\mathbf{I}(z) = 1|\mathbf{x})$, then MAE for the binary conditional expectation is obtained by summing the conditional probabilities corresponding to the value (0 or 1) not chosen for Ψ_{MAE}.

When the loss function l is defined by

$$l(\mathbf{I}(z), \Psi(\Xi(\mathbf{I}))(z)) = \begin{cases} 0, \text{ if } \begin{cases} \mathbf{I}(z) = \Psi(\Xi(\mathbf{I}))(z) = 0 \\ \textit{or} \\ \Psi(\Xi(\mathbf{I}))(z) = 1 \text{ and } P(\mathbf{I}(z) = 1|\mathbf{S} \cap W_z) = 1 \end{cases} \\ 1, \text{ otherwise} \end{cases} \tag{32}$$

it is called the *shape recognition* (*SR*) loss function. The operator Ψ_{SR} that minimizes the error associated with the shape recognition loss function is defined by

$$\Psi_{SR}(S)(z) = 1 \quad \Leftrightarrow \quad P(I(z) = 1 \mid S \cap W_z) = 1 \tag{33}$$

Under the assumption that $I(z) = 1$ if and only if a certain shape A is positioned at pixel z, this means that $\Psi_{SR}(S)(z) = 1$ if and only if A occurs at z given the observed random pattern $S \cap W_z$. If $x_1, x_2, \ldots, x_m$ denote the $m = 2^n$ possible realizations of $S \cap W_z$ and (without loss of generality) we let $x_1, x_2, \ldots, x_r$, $r < m$, be the patterns which when observed guarantee that the shape A occurs at z, then, as a Boolean function on realizations, $\Psi_{SR}(S)(z) = 1$ if and only if $S \cap W_z = x_j$, $j = 1, 2, \ldots, r$. Ψ_{SR} is called a *marker* because it identifies the shape characterized with certainty by the patterns $x_1, x_2, \ldots, x_r$. It may be that, the shape A occurs when $S \cap W_z = x_j$, $j > r$, but this is not certain because, for $j > r$,

$$P(I(z) = 1 \mid S \cap W_z = x_j) < 1 \tag{34}$$

Since the optimal operator is chosen from among W-operators, it can be canonically represented as a union of hit-or-miss transforms and optimization is over all operators of the form given in Eq. 22, where C_Ψ is a subset of the set of all canonical structuring pairs associated with W and where Ψ operates on random images. Owing to the restriction to canonical structuring pairs, $\Psi(S)(z) = 1$ if and only if there exists a pair $(E, F) \in C_\Psi$ such that $E = S \cap W_z$ and $F = S^c \cap W_z$. The class of structuring pairs defining the optimal filter Ψ_l is denoted by C_l and $(E, F) \in C_l$ if and only if

$$E[l(I(z), 1) \mid S \cap W_z = E] < E[l(I(z), 0) \mid S \cap W_z = E] \tag{35}$$

where $S \cap W_z = E$ means that the observed windowed process equals E and, because the structuring pair is canonical, $S^c \cap W_z = F$.

The structuring-pair classes defining the optimal operators Ψ_{MAE} and Ψ_{SR} for the MAE and SR loss functions are denoted by C_{MAE} and C_{SR}. For the MAE loss function,

$$E[l(I(z), 1) \mid S \cap W_z = E] = 1 - P(I(z) = 1 \mid S \cap W_z = E) \tag{36}$$

and, according to Eq. 35, $(E, F) \in C_{MAE}$ if and only if

$$P(I(z) = 1 \mid S \cap W_z = E) > 0.5 \tag{37}$$

For the SR loss function, $(E, F) \in C_{SR}$ if and only if

$$P(I(z) = 1 \mid S \cap W_z = E) = 1 \tag{38}$$

Since the hit-or-miss expansion corresponds to a logical sum of products, once the canonical expansion for the optimal operator has been determined, logic reduction can be employed to (nonuniquely) reduce the expansion. The result is a hit-or-miss expansion

with structuring pairs that are not necessarily canonical, but one which may involve significantly fewer logic gates.

5. Suboptimality

For various reasons, it is often necessary to employ a suboptimal (hopefully, close-to-optimal) filter Ψ_0 instead of the optimal filter Ψ_I. In terms of filter representation, this means that the expansion of Eq. 22 is taken over some structuring-element class $\mathbf{C}_0$ instead of $\mathbf{C}_I$. This increase in error can be quantified.

We define the *advantage* of a structuring pair (E, F) for which $P(\mathbf{S} \cap W_z = E) > 0$ by

$$\Delta_{I,(E,F)} = (E[I(\mathbf{I}(z), 0) \mid \mathbf{S} \cap W_z = E] - E[I(\mathbf{I}(z), 1) \mid \mathbf{S} \cap W_z = E])P(\mathbf{S} \cap W_z = E) \quad (39)$$

According to Eq. 35, $\Delta_{I,(E,F)} > 0$ if and only if $(E, F) \in \mathbf{C}_I$. There are two ways an increase in error can arise from using Ψ_0 instead of Ψ_I: $(E, F) \in \mathbf{C}_I$ but $(E, F) \notin \mathbf{C}_0$, or $(E, F) \notin \mathbf{C}_I$ but $(E, F) \in \mathbf{C}_0$. In either case, error increase from using Ψ_0 instead of Ψ_I is given by

$$\varepsilon\langle\Psi_0\rangle - \varepsilon\langle\Psi_I\rangle = \sum_{(E,F)\in\mathbf{C}_I\Delta\mathbf{C}_0} |\Delta_{I,(E,F)}| \quad (40)$$

where $\mathbf{C}_I \Delta \mathbf{C}_0$ is the symmetric difference between $\mathbf{C}_I$ and $\mathbf{C}_0$.

One way suboptimality is introduced is when there are too many structuring pairs for practical implementation, even after logic reduction. Focusing on canonical structuring pairs, these have been chosen according to Eq. 35. All contribute to a smaller error but some contribute very little. Those that contribute very little can be omitted from the expansion with a small loss in filter performance. If $\mathbf{C}_1 \subset \mathbf{C}_{MAE}$ and Ψ_1 is the filter corresponding to $\mathbf{C}_1$, then the increase in error owing to using Ψ_1 in place of Ψ_{MAE} is

$$\varepsilon\langle\Psi_1\rangle - \varepsilon\langle\Psi_{MAE}\rangle = \sum_{(E,F)\in\mathbf{C}_{MAE}-\mathbf{C}_1} \Delta_{I,(E,F)} \quad (41)$$

In practice, structuring pairs with positive advantage are listed in order from largest to smallest advantage. Filter logic is reduced by not including some number of structuring pairs at the bottom of the list prior to logic reduction.

For the MAE loss function, advantage is defined by

$$\Delta_{MAE,(E,F)} = (P(\mathbf{I}(z) = 1|\mathbf{S} \cap W_z = E) - P(\mathbf{I}(z) = 0|\mathbf{S} \cap W_z = E))P(\mathbf{S} \cap W_z = E) \quad (42)$$

In image restoration, $\Delta_{MAE,(E,F)}$ has been called the restoration effect of (E, F). For the SR loss function, advantage is defined by

$$\Delta_{SR,(E,F)} = \begin{cases} 1, & \text{if } P(\mathbf{I}(z) = 1|\mathbf{S} \cap W_z = E) = 1 \\ -P(\mathbf{I}(z) = 1|\mathbf{S} \cap W_z = E), & \text{otherwise} \end{cases} \quad (43)$$

6. Optimal Estimation of W-Operators

In practice, the optimal filter (operator) is estimated by an derived from realizations. Statistically, filter error $\varepsilon[\mathbf{I}, \Psi(\Xi(\mathbf{I}))]$ is estimated by taking realizations $I_1, I_2, \ldots, I_n$ of $\mathbf{I}$, applying Ξ to each realization to obtain corresponding observations, applying Ψ to obtain estimates, and then computing the average

$$\hat{\varepsilon}[\mathbf{I}, \Psi(\Xi(\mathbf{I}))] = \frac{1}{nM} \sum_{j=1}^{n} \sum_{z \in D} I(I_j(z) - \Psi(\Xi(I_j))(z)) \tag{44}$$

where the "hat" is used to indicate that the double empirical sum yields an estimate of the actual error and, owing to stationarity, the pixel z is irrelevant.

For realization-based design of the optimal filter in terms of the representation of Eq. 22 using the criterion of Eq. 35, the conditional expectations of Eq. 35 need to be estimated. Hence, for every structuring pair (E, F) we employ estimators

$$\hat{e}_{I,E}(0) = \hat{E}\,[I(\mathbf{I}(z), 0) \mid \mathbf{S} \cap W_z = E] \tag{45}$$

$$\hat{e}_{I,E}(1) = \hat{E}\,[I(\mathbf{I}(z), 1) \mid \mathbf{S} \cap W_z = E] \tag{46}$$

The designed "optimal" filter, $\hat{\Psi}_I$, is defined by Eq. 22 using the set $\hat{\mathbf{C}}_I$ of structuring pairs (E, F) for which

$$\hat{e}_{I,E}(1) < \hat{e}_{I,E}(0) \tag{47}$$

Relative to a given structuring pair (E, F), there is an increase in loss owing to estimation of the optimal filter if and only if the inequality of Eq. 35 holds but the inequality of Eq. 47 does not, or if Eq. 35 does not hold but Eq. 47 does. These cases correspond to two types of filter estimation error: $(E, F) \in \mathbf{C}_I$ but $(E, F) \notin \hat{\mathbf{C}}_I$; and $(E, F) \notin \mathbf{C}_I$ but $(E, F) \in \hat{\mathbf{C}}_I$. This is precisely the suboptimality situation covered by Eq. 40, where in the present case $\hat{\Psi}_I$ is the suboptimal filter being used in place of the optimal filter Ψ_I. Hence, the increase in error owing to estimation of the optimal filter from realizations is given by

$$\varepsilon\langle\hat{\Psi}_I\rangle - \varepsilon\langle\Psi_I\rangle = \sum_{(E,F) \in \mathbf{C}_I \Delta \hat{\mathbf{C}}_I} |\Delta_{I,(E,F)}| \tag{48}$$

For the MAE and SR loss functions, we can employ Eq. 35 or the respective equivalent conditions of Eqs. 37 and 38. For the latter, we can use the probability estimator

$$\hat{P}(\mathbf{I}(z) = 1 | \mathbf{S} \cap W_z = E) = \frac{\#(I(z) = 1 | S \cap W_z = E)}{\#(S \cap W_z = E)} \tag{49}$$

where the numerator gives the number of times the sample ideal images are 1-valued given (E, F) and the denominator gives the number times (E, F) is observed across the sample observed realizations. $(E, F) \in \hat{\mathbf{C}}_{MAE}$ or $(E, F) \in \hat{\mathbf{C}}_{SR}$ if and only if Eq. 37 or Eq. 38, respectively, holds with the estimate used in place of the actual conditional probability.

As stated, the error increase of Eq. 48 corresponds to a given designed structuring-pair class $\hat{\mathbf{C}}_l$. In fact, $\hat{\mathbf{C}}_l$ is a random collection depending on the realizations (randomly) selected. Thus, $\varepsilon\langle\hat{\Psi}_l\rangle - \varepsilon\langle\Psi_l\rangle$ is a random variable. From the perspective of statistical estimation, the derived filter estimates the actual optimal filter and the increase in filter error provides a measure of estimation goodness. Because this error increase is random, we can measure the *precision* of estimation by the expected error increase,

$$\varepsilon_l = E[\varepsilon\langle\hat{\Psi}_l\rangle - \varepsilon\langle\Psi_l\rangle] \tag{50}$$

Note that $(E, F) \in \mathbf{C}_l - \hat{\mathbf{C}}_l$ if and only if $\Delta_{l,(E,F)} > 0$ and $\hat{e}_{l,E}(1) \geq \hat{e}_{l,E}(0)$. Also $(E, F) \in \hat{\mathbf{C}}_l - \mathbf{C}_l$ if and only if $\Delta_{l,(E,F)} \leq 0$ and $\hat{e}_{l,E}(1) < \hat{e}_{l,E}(0)$. Hence,

$$\varepsilon_l = \sum_{(E,F)\in\mathbf{C}_l-\hat{\mathbf{C}}_l} \Delta_{l,(E,F)} P(\hat{e}_{l,E}(1) \geq \hat{e}_{l,E}(0)) - \sum_{(E,F)\in\hat{\mathbf{C}}_l-\mathbf{C}_l} \Delta_{l,(E,F)} P(\hat{e}_{l,E}(1) < \hat{e}_{l,E}(0)) \tag{51}$$

For the MAE loss function, this error has been studied in the context of binary-image restoration (Dougherty [12, 13]) and the analysis applies to the general binary theory. In particular, for the MAE loss function, Eq. 51 takes the form

$$\varepsilon_{MAE} = \tag{52}$$

$$\sum_{\{(E,F):P(\mathbf{I}(z)=1|\mathbf{S}\cap W_z=E)>0.5\}} |2P(\mathbf{I}(z)=1|\mathbf{S}\cap W_z=E)-1|P(\hat{P}(\mathbf{I}(z)=1|\mathbf{S}\cap W_z=E)\leq 0.5)P(\mathbf{S}\cap W_z=E)$$

$$+ \sum_{\{(E,F):P(\mathbf{I}(z)=1|\mathbf{S}\cap W_z=E)\leq 0.5\}} |2P(\mathbf{I}(z)=1|\mathbf{S}\cap W_z=E)-1|P(\hat{P}(\mathbf{I}(z)=1|\mathbf{S}\cap W_z=E)> 0.5)P(\mathbf{S}\cap W_z=E)$$

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

7. Machine Learning Theory and Optimal Operator Design

Computational learning theory (Anthony [1]) is considered to be 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, such as those currently used for training artificial neural nets. We briefly review basic elements of the model.

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 $P(\mathbf{B}|\mathbf{X})$, 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 will obtain a Boolean function, which will be able to answer if 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* $\mathbf{s}$ of length m is a sequence of m examples,

$$\mathbf{s} = ((x_1, b_1), (x_2, b_2), \ldots, (x_m, b_m)) \tag{53}$$

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 $\mathbf{s}$. 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 $\mathbf{S}$ of training samples is a random process with

$$P(\mathbf{S} = \mathbf{s}) = \prod_{i=1}^{m} P(\mathbf{X} = x_i) P(\mathbf{B} = b_i | x_i) \tag{54}$$

The probability mass for $\mathbf{S}$ 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 $\mathbf{s}$, for a target concept $t \in H$, an *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 $\mathbf{B}$ 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

$$r(h, t) = E[l(\mathbf{B}, h(\mathbf{X}))] \tag{55}$$

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

$$P(|r(L(\mathbf{S}), t) - r(h^*, t)| < \varepsilon) > 1 - \delta \tag{56}$$

for any target concept $t \in H$, distribution μ on D, and conditional distribution $P(\mathbf{B}|\mathbf{X})$. The value $m(\varepsilon, \delta)$ and the pair (ε, δ) are called, respectively, the *sample complexity* and *precision* of the learning algorithm.

If the size (number of elements) $\#(H)$ of the hypothesis space H is finite, then the sample complexity of any PAC learning algorithm L is bounded by

$$m(\varepsilon, \delta) = \frac{1}{\varepsilon^2}(log\ \#(H) + log\frac{1}{\delta}) \tag{57}$$

(Haussler [19]).

PAC learning theory was developed first by Valiant [33] for the case of consistent training samples and extended by Haussler [19], in a broader context, for nonconsistent training samples. The formulation we have presented is a simplification of the formulation of Haussler to concepts understood in the sense defined here. If the loss function is

$$l(\mathbf{B}, h(\mathbf{X})) = |\mathbf{B} - h(\mathbf{X})| \tag{58}$$

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

$$r(h, t) = \mu(\{x \in D: t(x) \neq h(x)\}) \tag{59}$$

and the formulation reduces to the definition of PAC learning algorithms in the original formulation of Valiant. Under these conditions, when the size of the hypothesis space is finite the bound for the sample complexity reduces to

$$m(\varepsilon, \delta) = \frac{1}{\varepsilon}(log\ \#(H) + log\frac{1}{\delta}) \tag{60}$$

The complete procedure of estimation of set operators that 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 $P(\mathbf{B}|\mathbf{X})$ results from nondeterminacy in the ideal image given an observed pattern and noise affecting the images.

Under the hit-or-miss representation of Eq. 22, for a hypothesis (Boolean function) h (determining a W-operator Ψ), a loss occurs if and only if $(E, F) \in \mathbf{C}_\Psi$, where (E, F) is the canonical structuring pair equivalent to $\mathbf{x}$, and $\mathbf{B} = 0$ when $\mathbf{x}$ is observed, or $(E, F) \notin \mathbf{C}_\Psi$ and $\mathbf{B} = 1$ when $\mathbf{x}$ is observed. Since $(E, F) \in \mathbf{C}_\Psi$ can be equivalently expressed as $\mathbf{x} \in \mathbf{C}_\Psi$ and $h(\mathbf{x}) = 1$ if and only if $\mathbf{x} \in \mathbf{C}_\Psi$,

$$\begin{aligned} r(h, t) &= E[l(\mathbf{B}, h(\mathbf{X}))] \\ &= \sum_{\mathbf{x}} E[l(\mathbf{B}, h(\mathbf{x}))|\mathbf{x}]P(\mathbf{X} = \mathbf{x}) \\ &= \sum_{\mathbf{x}\in\mathbf{C}_\Psi} E[l(\mathbf{B},1)|\mathbf{x}]P(\mathbf{X} = \mathbf{x}) + \sum_{\mathbf{x}\notin\mathbf{C}_\Psi} E[l(\mathbf{B},0)|\mathbf{x}]P(\mathbf{X} = \mathbf{x}) \end{aligned} \tag{61}$$

Assuming the target function t is a possible hypothesis,

$$r(h, t) - r(t, t) \tag{62}$$

$$= \sum_{\mathbf{x}\in\mathbf{C}_\Psi - \mathbf{C}_t} (E[l(\mathbf{B},1)|\mathbf{x}] - E[l(\mathbf{B},0)|\mathbf{x}])P(\mathbf{X} = \mathbf{x}) + \sum_{\mathbf{x}\in\mathbf{C}_t - \mathbf{C}_\Psi} (E[l(\mathbf{B},0)|\mathbf{x}] - E[l(\mathbf{B},1)|\mathbf{x}])P(\mathbf{X} = \mathbf{x})$$

where $\mathbf{C}_t$ 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. 61 applied to t, $E[l(\mathbf{B}, 1)|\mathbf{x}] \le E[l(\mathbf{B}, 0)|\mathbf{x}]$ if $\mathbf{x} \in \mathbf{C}_t$ and $E[l(\mathbf{B}, 0)|\mathbf{x}] \le E[l(\mathbf{B}, 1)|\mathbf{x}]$ if $\mathbf{x} \notin \mathbf{C}_t$. Hence, Eq. 62 reduces to

$$r(h, t) - r(t, t) = \sum_{\mathbf{x}\in\mathbf{C}_\Psi \Delta \mathbf{C}_t} |E[l(\mathbf{B},1)|\mathbf{x}] - E[l(\mathbf{B},0)|\mathbf{x}]| P(\mathbf{X} = \mathbf{x}) \tag{63}$$

which is equivalent to the suboptimality error-increase expression given in terms of absolute advantages in Eq. 40.

Relative to sampling, this equivalence involves Eq. 48. In these terms, the concept-learning precision inequality of Eq. 56 takes the form

$$P(|\varepsilon\langle\hat{\Psi}_l\rangle - \varepsilon\langle\Psi_l\rangle| < \varepsilon) > 1 - \delta \tag{64}$$

or

$$P\left(\sum_{(\varepsilon,F)\in C_i\Delta C_i}|\Delta_{l,(\varepsilon,F)}|<\varepsilon\right)>1-\delta \tag{65}$$

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

A theoretical contribution of machine learning theory is the sample complexity bound of Eq. 57; however, general machine-learning sample complexity bounds are unrealistically loose when compared with practical results found in the literature (Dougherty [13], Barrera [8]). In the next section, we will point out some reasons for this discrepancy.

8. Modeling Prior Information

Prior information 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 get a given precision (ε, δ) can become smaller or, equivalently, training samples of a fixed size can give sharper estimates.

For learning W-operators, a key point is the choice of a proper window W (Loce [24]). Window size should be as small as possible since the size of the domain D is affected exponentially by the size of W, $\#(D) = 2^{\#(W)}$. In this task, prior information plays a salient role. Usually 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 (Kong [22]) depends just on a small neighborhood of 4 or 8 pixels, depending on the connectivity required; if the target concept is a filter of connected components or holes, then shape recognition can be described trivially by canonical hit-miss operators that depend on the smallest window that contain all the components to be filtered.

An important factor related to window size is image resolution. Window size should be proportional to image resolution: higher (sharper) 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 $r << n$. 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 yields Nn^2 examples (observations) and, owing to repititions, 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 at hand, 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 solve an imaging problem is a basic preliminary step in automatic programming of MMach's.

In applications of W-operators to solve practical image analysis problems, we are almost always restricted to specific contexts, that is, to particular classes of images for which the operators should perform properly. There are many examples of common contexts: images of a family of similarly printed circuits, images of texts printed by the same printer or line of printers, microscopic images of particular materials or cell cultures, images of ZIP codes and addresses written on letters, images of objects under industrial automatic quality control, etc. To restrict to a given context implies that the domain D

becomes a subset of the power set of W. Usually, it is not easy to estimate the number of patterns in a given context, but it is often significantly smaller than $2^{|W|}$.

The worst probabilistic structure for the domain is the uniform distribution, which models the most disorganized space. This distribution is bad for concept learning because to get small risks for an 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 extreme situations and are usually far from both.

For certain types of applications, such as noise restoration in document processing (Baird [2]) or texture analysis in material science (Serra [31]), good stochastic models for describing image classes are known. In these cases, besides the domain being restricted by the context, its distribution is known from the stochastic model. Knowledge of the probabilistic structure of the domain may be used to guide the choice of the training sample: the size of the training sample must be sufficient to cover the most significant part of the domain. In other situations, a large quantity of data is available and it is possible to make good estimations of the most significant parts (where the probability mass is concentrated) of the distribution. The computational cost of estimating the significant parts of a distribution is low, relatively to the cost of estimating a concept from a learning algorithm using the same amount of data. Hence, a large amount of data may be used to estimate the distribution and just a portion of data be chosen for the training samples.

Information concerning properties of the target operator is also useful. If the target operator is a marker for shape recognition, it is antiextensive; if the target operator is a size classifier, it is increasing, antiextensive and idempotent; etc. These kind of properties can be interpreted as constraints that characterize families of operators in the hypothesis space (the increasing operators, the antiextensive operators, etc.). Under the knowledge of target-operator properties, the hypothesis space considered will be the intersection of the families of operators defined by the constraints. This is an obvious procedure. Of course, if we know that an operator is antiextensive, there is no reason to look for it among the extensive operators. Specialized algorithms for learning increasing operators have been developed (Loce [23-25], Han [17], Mathew [27]). In addition, estimation precision under the family of increasing operators has been studied, in particular, the significant decrease in sampling size for the increasing class (Dougherty [13]).

A last kind of information that should be considered is knowledge of a good initial hypothesis for the target operator. In this case, 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. Dougherty and Loce [12] have used this technique to design operators for restoration of text images. 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. All types of prior information discussed have in common the fact

that they reduce the number of subsets of W with unknown label. Therefore, they imply in smaller sample complexities. The choice of a small window reduces domain size. Context can determine subsets of W that should be labeled; others will be taken as don't-cares. Dstributional knowledge allows the 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 — for instance, when learning an increasing operator, if a pattern is labeled 1, then all patterns including it must also be labeled 1. Finally, an initial hypothesis may label a large number of patterns.

9. Detailed design procedure

The design procedure of set operators for binary image analysis is composed of two main steps: (1) estimation of conditional probabilities; (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 observation window to all pixel locations within the observation image;
2. At each location, record the observed canonical template.
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 template, tally the number of times a one was observed in the ideal image and the number of times a zero was observed.
5. For the operator representation select, for each canonical template 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.

Simplification of a Boolean function consists of transforming a given expression into another expression with less terms or literals. A Boolean expression is considered a *minimal expression* if: (1) there does not exist an equivalent Boolean expression with a smaller number of terms; (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 according to laws and theorems of Boolean algebra. We consider some options to perform the logical minimization.

In order to present these options, we recall some basic definitions. Let W be a finite set and $P(W)$ be its power set (set of all subsets of W). Earlier we defined an interval in terms of binary functions on W. Because a binary function can be treated a subset (the set of all 1-valued elements), the interval definition can be reformulated in terms of substs: if $A, B \in P(W)$ and $A \subset B$, an *interval*, or *cube*, $[A, B]$ in $P(W)$ is the collection of all sets X such that $A \subset X \subset B$. The *dimension* of a cube $[A, B]$ is defined by

$$dim([A,B]) = \#(B) - \#(A) \tag{66}$$

A cube of dimension n is called an n-cube.

9.1. Quine-McCluskey minimization

A classical simplification algorithm for Boolean expressions is the Quine-McCluskey tabular algorithm (Quine [29], McCluskey [28]). This algorithm provides a minimal expression and is well-suited for computer implementation since it systematizes the simplification process. The Quine-McCluskey 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 illustrates the generation of the set of prime implicants in a simple example. The drawback with the Quine-McCluskey 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. In practice, this algorithm cannot be used for functions with more than twelve variables.

9.2. ISI learning algorithm

The *Incremental Splitting of Intervals* (*ISI*) algorithm has been proposed (Barrera [7]) to deal with functions that have a large number of variables and a relatively small number of fixed values, a common occurrence in the learning of set operators. The main idea under the Quine-McCluskey algorithm is repetition of a process that joins cubes (from the 0-cubes) to get cubes as large as possible. The main idea under 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 T be an initial set of prime implicants. Usually, in a space of dimension n, T is the set whose single element is the n-cube. If there are no negative examples, then the algorithm stops and T itself is the output.
2. Separate the cubes in T according to the following criterion: put the cubes of T that cover the next negative example to be extracted in N and the others in P. Set $M = \varnothing$.
3. Split each cube in N and represent it by its low order cubes. The low order cubes that were not covered by some cube in P are put in M.
4. Put $T = P \cup M$ and repeat steps 1 through 3 for the next negative example, while negative examples exist.

5. Select the essential prime implicants (by the same method used in the QuineMcCluskey algorithm).

A fundamental aspect of the ISI algorithm is splitting a cube by a negative example. A cube $[A, B] \subset P(W)$ is *split* by a negative example $X \in [A, B]$ into a collection of intervals, the collection denoted by $[A, B] - X$, in the following manner

$$[A, B] - X = \{[A, B \cap \{a\}^c : a \in X\} \cup \{[A \cup \{b\}, B] : b \in X^c\} \tag{67}$$

where the complement is taken with respect to W. It can be proved that

$$\#([A, B] - X) = dim([A, B]) \tag{68}$$

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 $[A_1, B_1] \in [A, B] - X$,

$$dim([A_1, B_1]) = dim([A, B]) - 1 \tag{69}$$

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 $[A, B] - X$ are maximal.

The ISI algorithm is recursive. Each execution of step 4 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 Figure 2. The same process can be viewed in a 3-space in Figure 3.

Table 1 shows comparative experimental results between the Quine-McCluskey and ISI algorithms for an example with twelve variables. The columns "negative", "positive" and "d.c." indicate the numbers of distinct negative examples, positive examples, and don't cares, respectively; the column terms indicate the number of terms in the minimal expression; "mem" indicates the maximal quantity of cubes that simultaneously exist during the minimization process. Table 2 gives experimental results of the application of the ISI algorithm for minimization of functions with large numbers of variables that are not treatable by the Quine-McCluskey algorithm. These experiments have been performed using an INTEL-586 processor. Processing time is measure in hours (h), minutes (m) and seconds (s). From Table 1, we conclude that application of the ISI algorithm uses much less storage space than the QM algorithm and that efficiency depends on the quantity and distribution of the examples given. Table 2 shows that the ISI algorithm can be applied for minimizing expressions with large numbers of variables, so long as the number of don't cares is also large.

10. Application examples

In this section, we present some example applications of optimal design of set operators in binary image analysis using the ISI learning algorithm. Example 4 employs the *SR* loss function; all others use the MAE loss function.

Example 1: *Edge detection* - The window is the 3 × 3 square and the training sample is consistent. The training sample is taken from the images of Figure 4a. The size of the training sample 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:

$$\begin{bmatrix} 0 & x & x \\ x & 1 & x \\ x & x & x \end{bmatrix}, \begin{bmatrix} x & x & 0 \\ x & 1 & x \\ x & x & x \end{bmatrix}, \begin{bmatrix} x & x & x \\ x & 1 & x \\ x & x & 0 \end{bmatrix}, \begin{bmatrix} x & x & x \\ x & 1 & x \\ 0 & x & x \end{bmatrix}$$

Figure 4b shows application of the learned operator.

Example 2: *Edge Detection in Noise* - The window is the 3 × 3 square. The training sample of size 80,392 was taken from the images of Figure 5a. There were 363 distinct examples observed, 278 negative and 85 positive. Learning time was 1s The resulting basis is composed by 44 intervals. The noise is punctual, additive and subtractive, and uniformly distributed, with density 5% The error measure was 0.48%. Figure 5b shows application of the learned operator.

Example 3: *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 Figure 6a. 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:

$$\begin{pmatrix} x & 0 & 1 & 0 & x \\ 0 & x & 1 & 0 & 0 \\ 0 & x & 1 & 0 & x \\ x & 0 & 0 & x & x \\ x & x & x & x & x \end{pmatrix}, \begin{pmatrix} x & 0 & 0 & 0 & x \\ 0 & x & x & 0 & x \\ 1 & 1 & 1 & 0 & x \\ x & x & x & 0 & x \\ x & x & x & x & x \end{pmatrix}, \begin{pmatrix} x & x & 0 & x & x \\ x & 0 & 0 & x & x \\ x & 0 & 1 & 1 & 1 \\ x & 0 & 0 & x & x \\ x & x & x & x & x \end{pmatrix}, \begin{pmatrix} x & x & x & x & x \\ 0 & 0 & 0 & 0 & 0 \\ 0 & x & 1 & 0 & x \\ 0 & x & 1 & x & x \\ x & x & 1 & x & x \end{pmatrix}$$

Note that, since the corners are not part of the window, each corner pixel in the basis mudy be a don't-care pixel. The noise is punctual, additive and subtractive, and uniformly

distributed, with densities 1% and 0.1%. The error measure was 0.005%. Figure 6b shows application of the learned operator.

Example 4: *Letter Recognition* - The window is the 3 × 3 square. The training sample of size 24,426 was taken from the images of Figure 7a. There were 273 distinct observed examples, 271 negatives and 2 positives. Learning time was 1s. The resulting basis is composed of 2 intervals. The error measure was zero. Figure 7b shows application of the learned operator. For a more sophisticated OCR application, see Barrera [9].

Example 5: *Texture Recognition* - The window is the 5 × 5 square. The training sample of size 90,126 was taken from the images of Figure 8a. There were 5,098 distinct observed examples, 5,024 negatives and 74 positives. Learning time was 1s. The resulting basis is composed by 12 intervals. Figure 8b shows application of the learned operator.

Example 6: *Image Restoration* - The window is the 3 × 3 square. The training sample of size 48,852 was taken from the images of Figure 9a. There were 482 distinct observed examples, 247 negative and 222 positive. Learning was 1s. The resulting basis is composed by 28 intervals. The noise is punctual, additive and subtractive, and uniformly distributed, with density 5%. The error measured was 1.2%. Figure 9b shows the result of the application of the operator learned. We have included this example only for completeness. Optimally designed hit-or-miss-based algorithms have been studied extensively by Loce [23]. The results obtained here are consistent with those expected from a small window; better results are obtained by using a larger window.

Example 7: *Defect Lines* - Detection of defect lines in eutectic alloys is a classical problem in mathematical morphology. A first ad hoc solution for this problem was given by Christian Lantejoul (Serra [31]) and this problem was subsequently addressed by Schmitt [30]. The goal is to find defect lines in an image of a transversal section of an eutictic alloy. We have applied our approach of designing set operators to this problem. At first, we have performed a homotopic transformation on the image and a shrink in such a way that we can observe the defect lines in a low resolution image. The transformed image has been divided into two halfs, the first to be used as training data and the second for testing the result. Figure 10a shows the images used as training data and Figure 10b shows application of the operator. We should remark that the desgined operator is not rotation invariant as are the operators proposed by Lantejoul and Schmitt; however, the results can be considered quite good if one takes into account the small amount of data available for training. The window was the 5 × 5 square. 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.

Example 8: *Barcode Segmentation* The goal in this application is to separate the bar code region in document images. The images were acquired by a scanner, thresholded, closed by a large vertical segment, and strongly shrunk before been used in the automatic programming. The training phase used 4 pairs of images and an 11 × 11 window (121 Boolean variables). The size of the training sample was 20,764 with 13,067 distinct examples, 11,659 negatives and 1,408 positives. The time of learning was 1m28s. The

resulting basis is composed of 23 intervals. The error measured was 0.035%. Figure 11a shows a pair of images used in trainning and Figure 11b shows an application of the learned operator. It is of interest to note that we have used a large window and, even with the small amount of examples used, the precision is quite good.

11. Conclusion

This paper has presented a generalization of the formulation proposed by Dougherty for the design of set operators by statistical optimization in the context of morphological representations. The original formulation deals just with the problem of restoration and (later) enhancement of binary images; the new one provides a general approach to the problem of designing set operators for MMachs. This formalism gives a solid mathematical structure to the design morphological operators from examples, a topic initially approached by Vogt [34] in heuristic terms. It has also been demonstrated that the new formulation can be interpreted as a restriction of the theory proposed by Haussler for the generalization of the concept of PAC learning algorithms (proposed by Valiant). Haussler understands concepts as functions on a real interval, while the restriction considered takes concepts as Boolean functions. Reasons have been given why the analytical bounds for sample complexity given by PAC learning are quite unrealistically loose in practice. The ISI algorithm for reducing W-operators in canonical form has been discussed and applied, along with the entire procedure, to image analysis problems.

The mathematical fundamentals applied in the proposed methodology are canonical operator representation by mathematical morphology and statistical optimization. Because statistical optimization is a quite general theory and any complete lattice operator can be represented in a canonical form (Banon [4], Dougherty [15]) the present formulation can be extended to other application domains such as signal processing, gray-scale and colored image processing (Dougherty [14]). The theory has a natural counterpart in the theory of supervised neural networks, since both methodologies involve training via input-output image pairs with the result being a designed set operator. An essential difference between neural networks and morphological machines is the use of networks of artificial neurons in the former as opposed to morphological operator representation in the latter.

12. References

1. Anthony, M., and N. Biggs, *Computational Learning Theory – An Introduction.* Cambridge University Press, 1992.

2. Baird, H. S., "Document Image Defect Models and Their Uses," *Proc. IAPR 2nd Int'l Conf. on Document Analysis and Recognition*, Tsukuba Science City, October, 1993.

3. Banon, G. J. F., and J. Barrera, "Minimal Representation for Translation Invariant Set Mappings by Mathematical Morphology," *SIAM J. Applied Mathematics*, **51**, pp. 1782-1798, 1991.

4. Banon, G. J. F., and J. Barrera, "Decomposition of Mappings Between Complete Lattices by Mathematical Morphology, Part I. General Lattices," *Signal Processing*, **30**, pp. 299-327, 1993.

5. Barrera, J., and G. J. F. Banon, "Expressiveness of the Morphological Language," *SPIE Image Algebra and Morphological Image Processing*, **1769**, pp. 264-275, San Diego, 1992.

6. Barrera, J., Banon, G. J. F., and R. A. Lotufo, "A Mathematical Morphology Toolbox for the KHOROS System," *SPIE Image Algebra and Morphological Image Processing*, **2300**, pp. 229-240, San Diego, 1994.

7. Barrera, J., Tomita, N. S., and F. S. Côrrea da Silva, "Automatic Programming of Morphological Machines by PAC Learning," *SPIE Neural, Morphological and Stochastic Methods in Image and Signal Processing*, **2568**, pp. 233-244, San Diego, 1995.

8. Barrera, J., Salas, G. P., and R. F. Hashimoto, "Set Ooperations on Collections of Closed Intervals and Their Applications to the Automatic Programming of Mmach's," *Proc. ISMM '96*, Atlanta, 1996.

9. Barrera, J., Terada, R., Côrrea da Silva, F. S., and N. S. Tomita, "Automatic Programming of Morphological Machines for OCR," *Proc. ISMM '96*, Atlanta, 1996.

10. Dougherty, E. R., "Optimal Mean-Square *N*-Observation Digital Morphological Filters – Part I: Optimal Binary Filters," *CVGIP: Image Understanding*, **55** (1), pp. 36-54, 1992.

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), pp. 815-823, 1993.

12. 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.

13. Dougherty, E. R., and R. P. Loce, "Precision of Morphological-Representation Estimators for Translation-Invariant Binary Filters: Increasing and Nonincreasing," *Signal Processing*, **40** (3), pp. 129-154, 1994.

14. Dougherty, E. R., and D. Sinha, "Computational Mathematical Morphology," *Signal Processing*, **38**, 1994.

15. Dougherty, E. R., and D. Sinha, "Computational Gray-Scale Mathematical Morphology on Lattices (A Comparator-based Image Algebra) — Part I: Architecture," *Real-Time Imaging*, **1** (1), 1995.

16. Goutsias, J., "Morphological Analysis of Discrete Random Shapes, *Mathematical Imaging and Vision*, **2** (2/3), pp. 193-215, 1992.

17. 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, pp. 41-44, 1994.

18. Hassoun, M. H., *Fundamentals of Artificial Neural Networks*, MIT Press, 1995.

19. Haussler, D., "Decision Theoretic Generalizations of the PAC Model for Neural Nets and Other Learning Applications. *Information and Computation*, 100, pp. 78-150, 1992.

20. Klein, J., and J. Serra, "The Texture Analyser," *Microscopy*, 95, pp. 343-356, 1972.

21. Klein, J., and R. Peyhard, "PIMM1, An Image Processing ASIC Based on Mathematical Morphology," *IEEE ASIC Seminar and Exibit*, pp. 25-28, Rochester, 1989.

22. Kong, T. Y., and A. Rosenfeld, "Digital Topology: Introduction and Survey," *CVGIP*, 48, pp. 357-393, 1989.

23. Loce, R. P., and E. R. Dougherty, *Enhancement and Restoration of Digital Documents: Statistical Design of Nonlinear Algorithms*, SPIE Press, Bellingham, 1996.

24. 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.

25. 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.

26. Matheron, G. *Random Sets and Integral Geometry*. Wiley, New York, 1975.

27. 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) pp. 409-430, 1993.

28. McCluskey, E. J., " Minimization of Boolean Functions," *Bell System Tech. J.*, 35 (5), pp. 1417-1444, 1956.

29. Quine, W. V., "The Problem of Simplifying Truth Functions," *Am. Math. Monthly*, 59 (8), pp. 521-531, 1952.

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

31. Serra, J., *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.

32. Serra, J., *Image Analysis and Mathematical Morphology, Volume 2: Theoretical Advances*, Academic Press, New York, 1989.

33. Valiant, L. "A theory of the learnable," *Comm. ACM*, 27, pp. 1134-1142, 1984.

34. Vogt, R., *Automatic Generation of Morphological Set Recognition Operators*, Springer-Verlag Inc., New York, 1989.

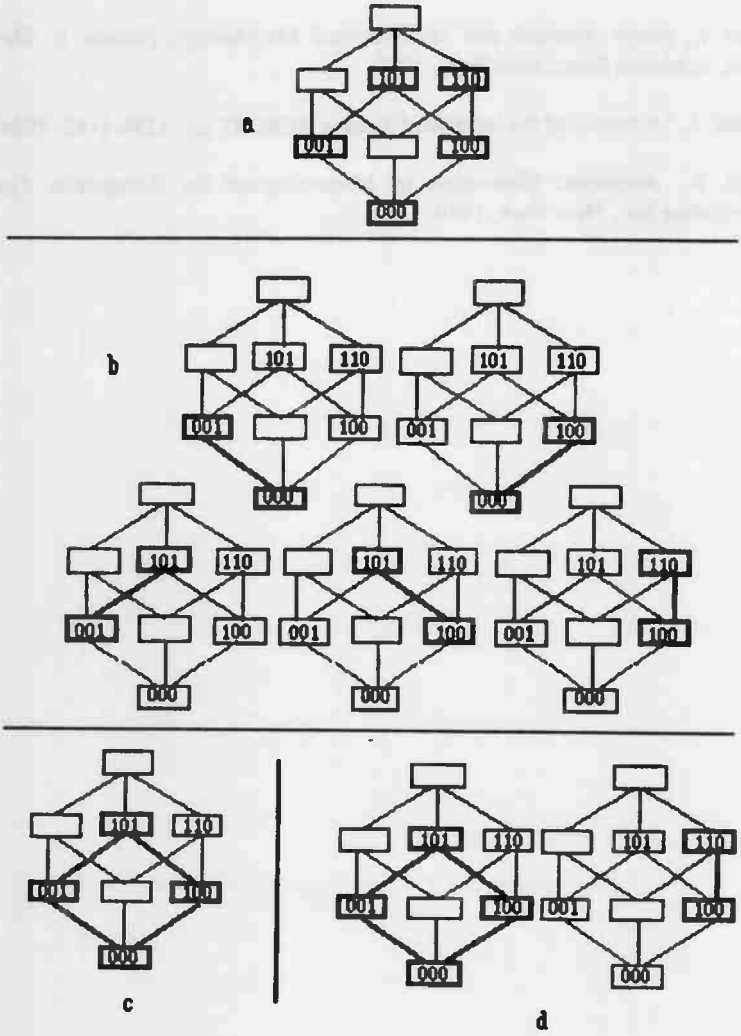


Figure 1

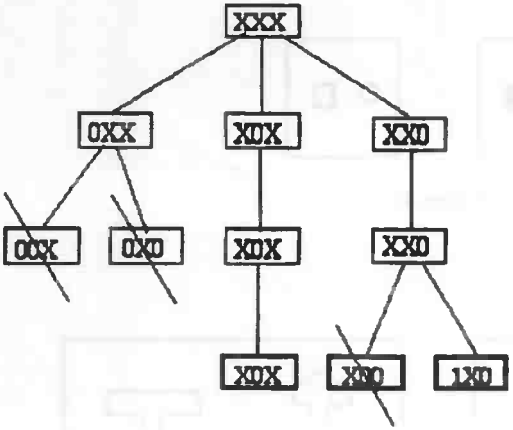


Figure 2

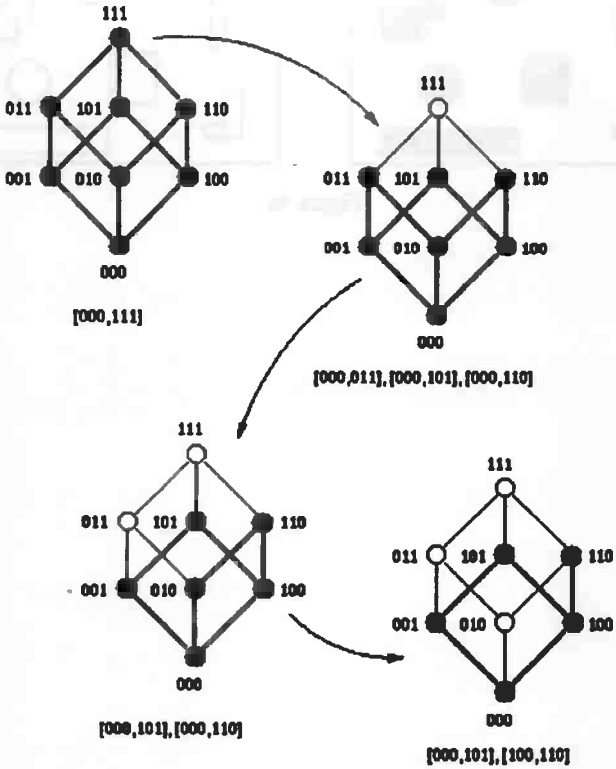


Figure 3

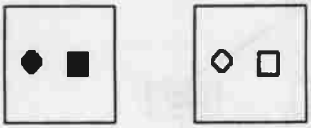


Figure 4a

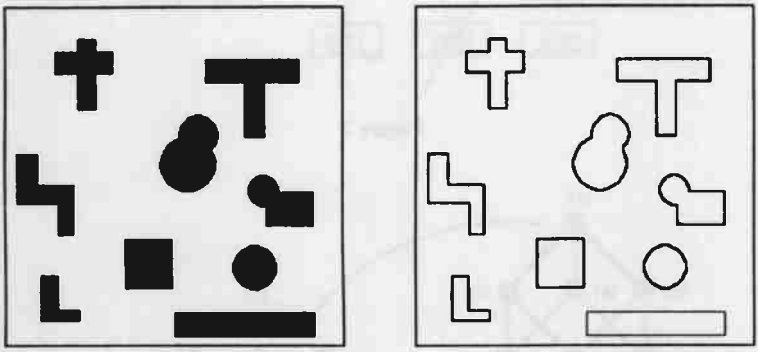


Figure 4b

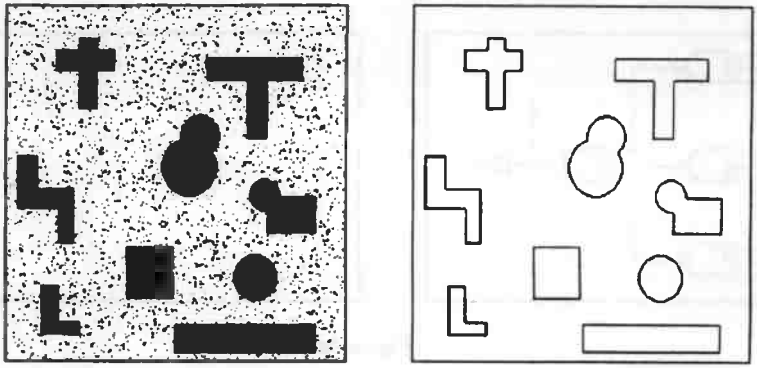


Figure 5a

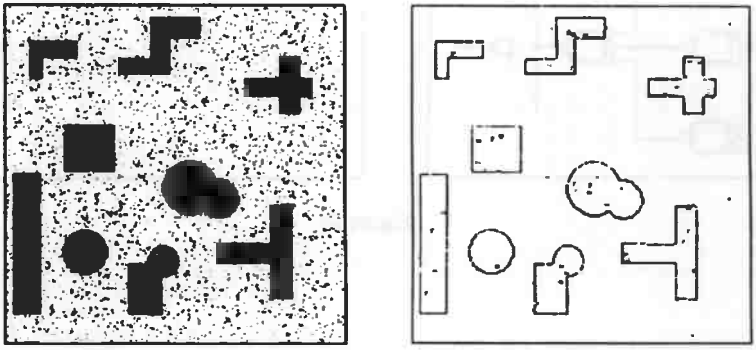


Figure 5b

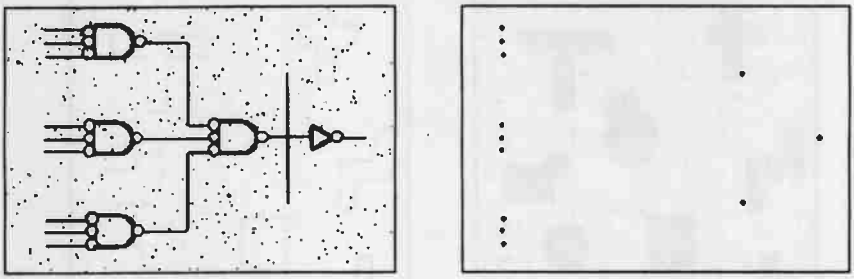


Figure 6a

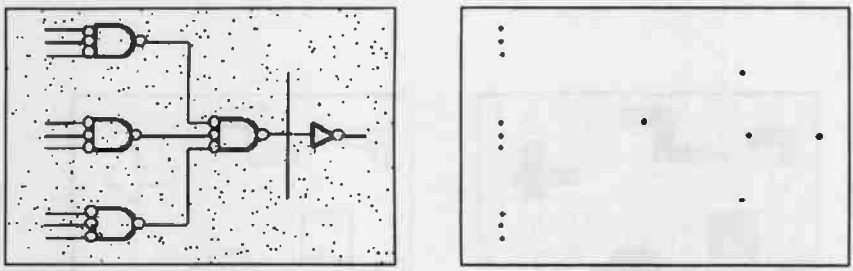


Figure 6b

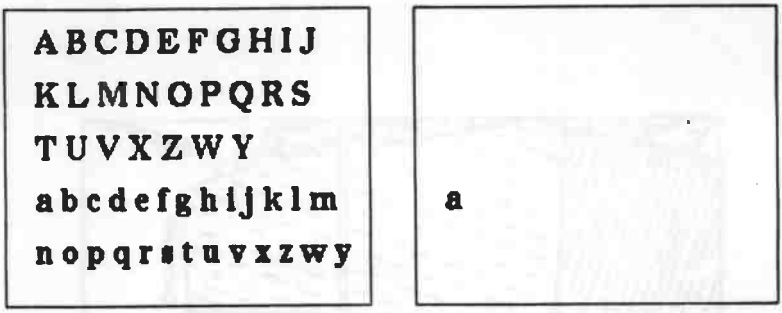


Figure 7a

CHARLES BABBAGE Everyone from bankers to navegators depended on mathematical tables during the Industrial Revolution. However, these hand-calculated tables were usually full of erros. After discovering that his own tables were riddled with mistakes. Charles Babbage envisioned a steam powered differential engine and then an analytical engine that would perform tedious calculations accurately.

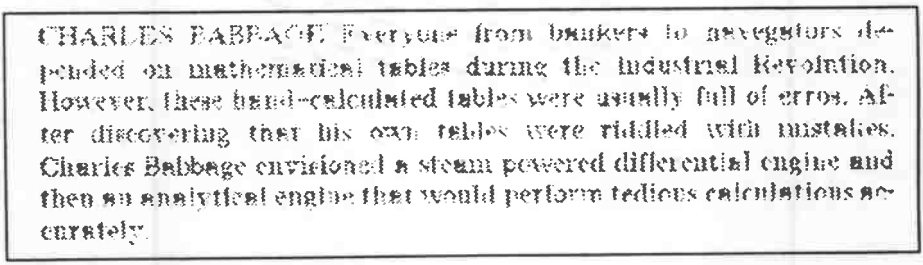


Figure 7b

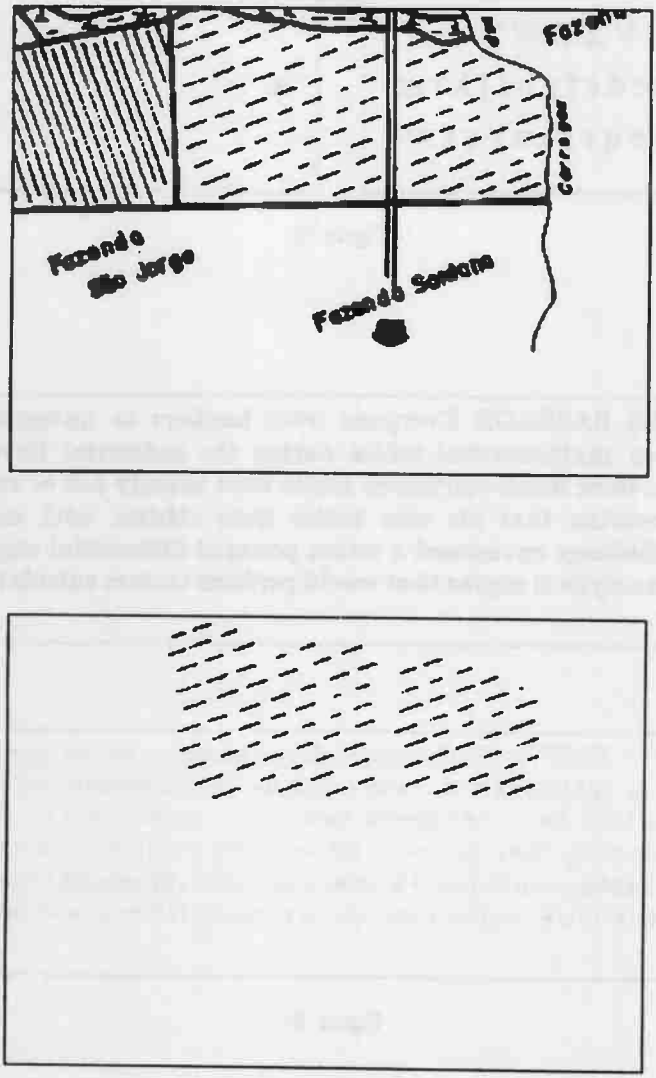
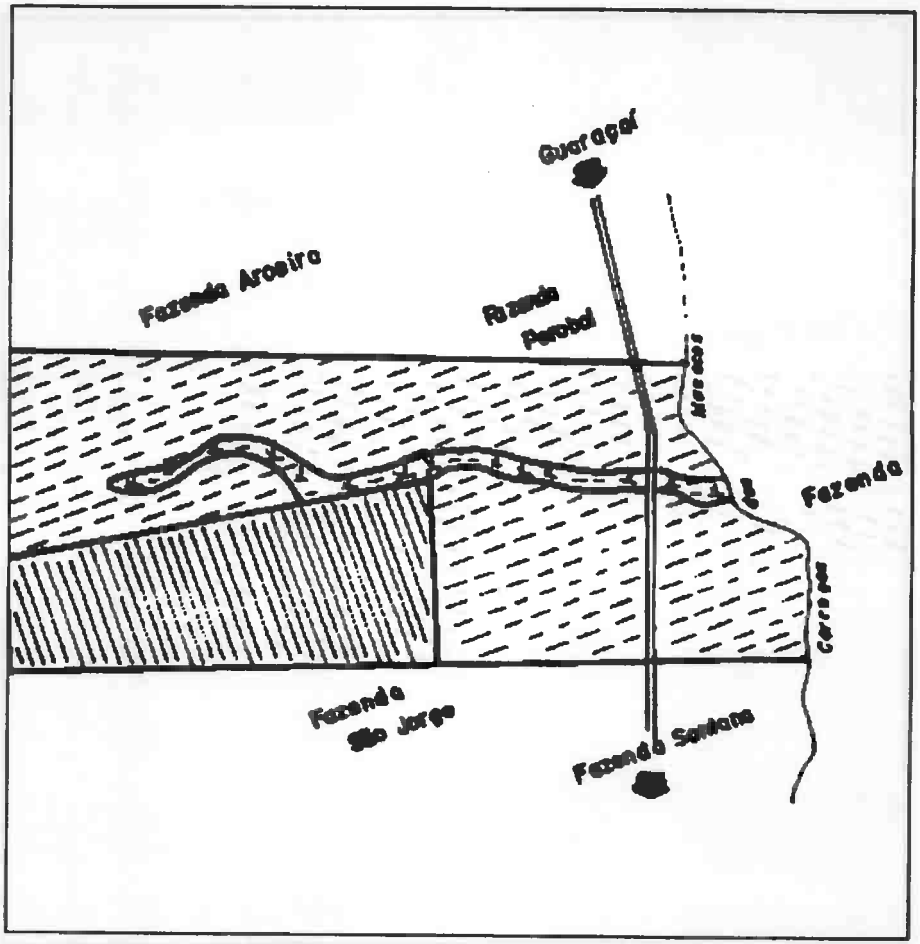


Figure 8a


Guaraçaí
Fazenda Arosira
Fazenda
Fazenda
Fazenda
São Jorge
Fazenda Santana

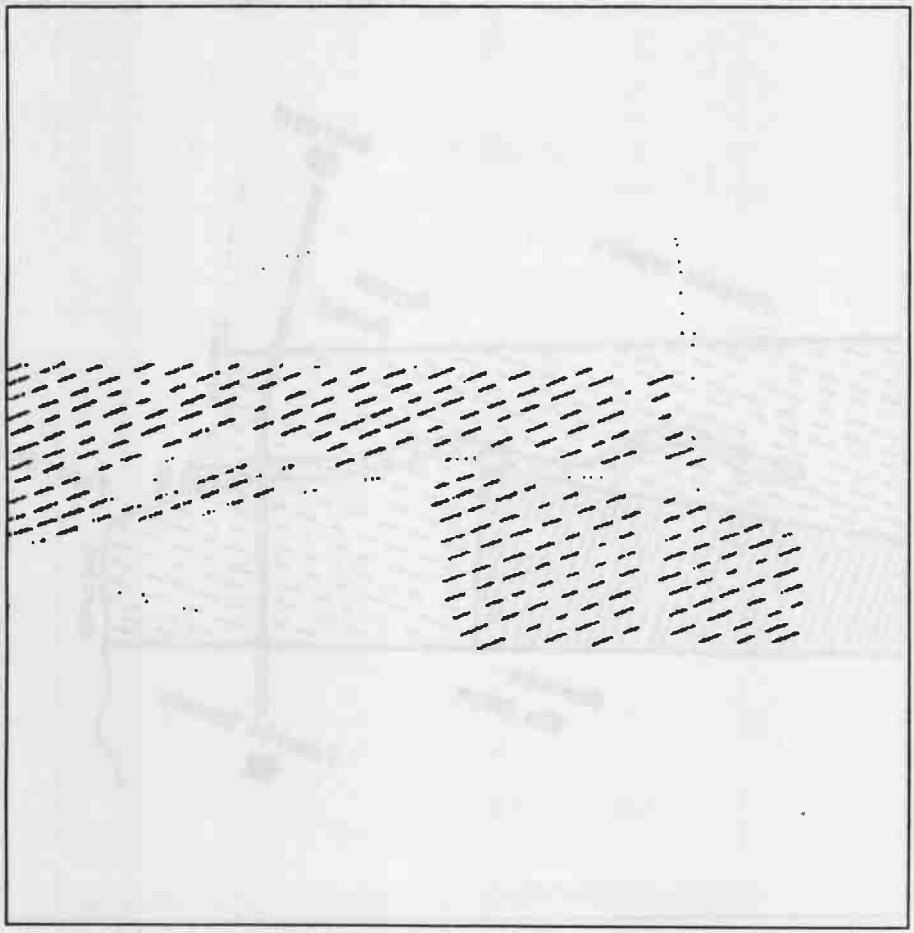


Figure 8b

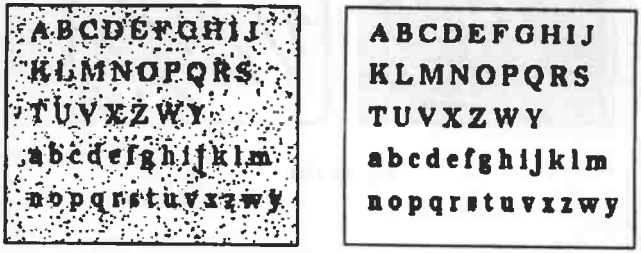


Figure 9a

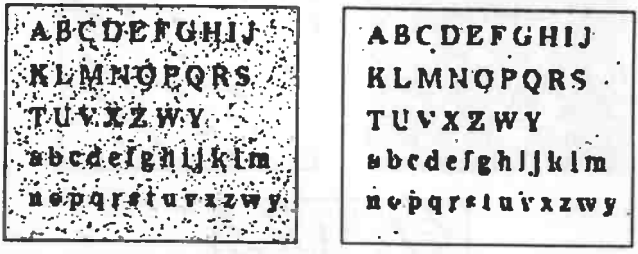


Figure 9b

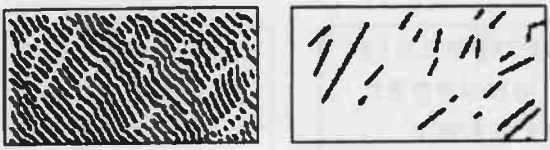


Figure 10a

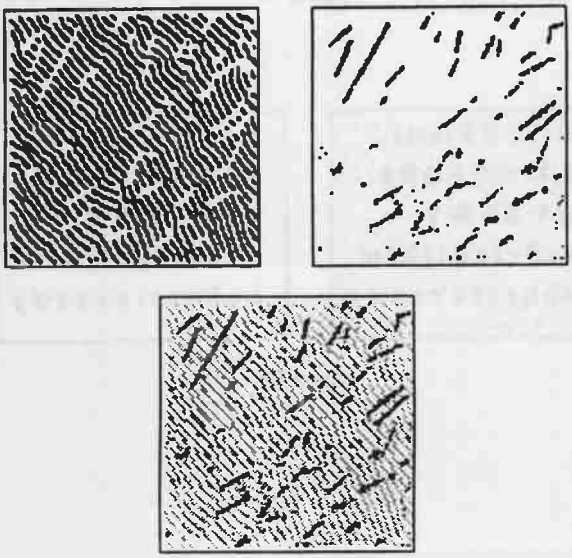


Figure 10b

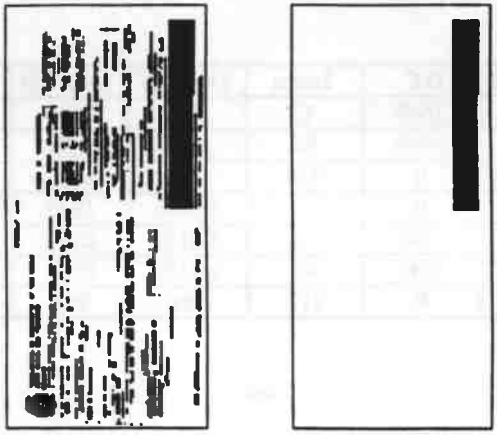


Figure 11a

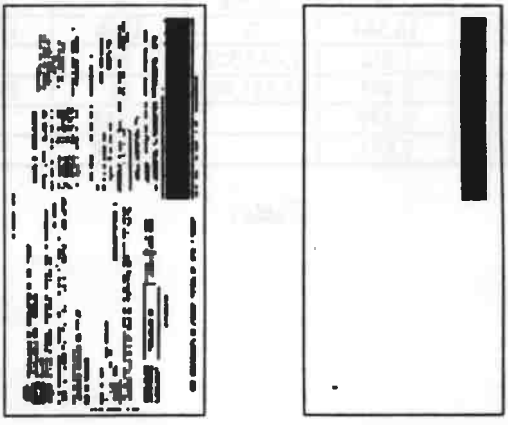


Figure 11b

Negatives	Positives	D.C.	Terms	Time-ISI	Mem-ISI	Time-QM	Mem-QM
835	601	2660	82	21s	2160	16m	80682
712	724	2660	90	24s	2202	20m10s	89824
1035	3061	0	430	7m	4580	7m7s	38524
2048	2048	0	418	1m30s	1942	1m22s	15876
2467	4823	1147	122	3m33s	2804	15s	11902
2859	90	1147	40	2m52s	2525	6s	7807
3061	1035	0	411	26s	993	14s	3069

Table 1

Variables	Negatives	Positives	D.C.	Terms	Memory	Time
15	42	18	32,708	6	4146	29s
15	16,384	16,384	0	1768	12,728	6h46m14s
20	21	1,052	1,047,503	25	744	9s
25	36	2,493	33,551,903	32	32,880	1h42m1s
49	56,191	4,239		1,348		1h46m15s
81	67,119	7,823		2,798		14h42m39s

Table 2

List of Figures and Tables

RELATÓRIOS TÉCNICOS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Instituto de Matemática e Estatística da USP

A listagem contendo os relatórios técnicos anteriores a 1993 poderá ser consultada ou solicitada à Secretaria do Departamento, pessoalmente, por carta ou e-mail(mac@ime.usp.br).

Imre Simon
THE PRODUCT OF RATIONAL LANGUAGES
RT-MAC-9301, Maio 1993, 18 pp.

Flávio Soares C. da Silva
AUTOMATED REASONING WITH UNCERTAINTIES
RT-MAC-9302, Maio 1993, 25 pp.

Flávio Soares C. da Silva
ON PROOF-AND MODEL-BASED TECHNIQUES FOR REASONING WITH UNCERTAINTY
RT-MAC-9303, Maio 1993, 11 pp.

Carlos Humes Jr., Leônidas de O.Brandão, Manuel Pera Garcia
A MIXED DYNAMICS APPROACH FOR LINEAR CORRIDOR POLICIES
(A REVISITATION OF DYNAMIC SETUP SCHEDULING AND FLOW CONTROL IN MANUFACTURING SYSTEMS)
RT-MAC-9304, Junho 1993, 25 pp.

Ana Flora P.C.Humes e Carlos Humes Jr.
STABILITY OF CLEARING OPEN LOOP POLICIES IN MANUFACTURING SYSTEMS (Revised Version)
RT-MAC-9305, Julho 1993, 31 pp.

Maria Angela M.C. Gurgel e Yoshiko Wakabayashi
THE COMPLETE PRE-ORDER POLYTOPE: FACETS AND SEPARATION PROBLEM
RT-MAC-9306, Julho 1993, 29 pp.

Tito Homem de Mello e Carlos Humes Jr.
SOME STABILITY CONDITIONS FOR FLEXIBLE MANUFACTURING SYSTEMS WITH NO SET-UP TIMES
RT-MAC-9307, Julho de 1993, 26 pp.

Carlos Humes Jr. e Tito Homem de Mello
A NECESSARY AND SUFFICIENT CONDITION FOR THE EXISTENCE OF ANALYTIC CENTERS IN PATH FOLLOWING METHODS FOR LINEAR PROGRAMMING
RT-MAC-9308, Agosto de 1993

Flavio S. Corrêa da Silva
AN ALGEBRAIC VIEW OF COMBINATION RULES
RT-MAC-9401, Janeiro de 1994, 10 pp.

Flavio S. Corrêa da Silva e Junior Barrera
AUTOMATING THE GENERATION OF PROCEDURES TO ANALYSE BINARY IMAGES
RT-MAC-9402, Janeiro de 1994, 13 pp.

Junior Barrera, Gerald Jean Francis Banon e Roberto de Alencar Lotufo
A MATHEMATICAL MORPHOLOGY TOOLBOX FOR THE KHOROS SYSTEM
RT-MAC-9403, Janeiro de 1994, 28 pp.

Flavio S. Corrêa da Silva
ON THE RELATIONS BETWEEN INCIDENCE CALCULUS AND FAGIN-HALPERN STRUCTURES
RT-MAC-9404, abril de 1994, 11 pp.

Junior Barrera; Flávio Soares Corrêa da Silva e Gerald Jean Francis Banon
AUTOMATIC PROGRAMMING OF BINARY MORPHOLOGICAL MACHINES
RT-MAC-9405, abril de 1994, 15 pp.

Valdemar W. Setzer; Cristina G. Fernandes; Wania Gomes Pedrosa e Flavio Hirata
UM GERADOR DE ANALISADORES SINTÁTICOS PARA GRAFOS SINTÁTICOS SIMPLES
RT-MAC-9406, abril de 1994, 16 pp.

Siang W. Song
TOWARDS A SIMPLE CONSTRUCTION METHOD FOR HAMILTONIAN DECOMPOSITION OF THE HYPERCUBE
RT-MAC-9407, maio de 1994, 13 pp.

Julio M. Stern
MODELOS MATEMATICOS PARA FORMAÇÃO DE PORTFÓLIOS
RT-MAC-9408, maio de 1994, 50 pp.

Imre Simon
STRING MATCHING ALGORITHMS AND AUTOMATA
RT-MAC-9409, maio de 1994, 14 pp.

Valdemar W. Setzer e Andrea Zisman
*CONCURRENCY CONTROL FOR ACCESSING AND COMPACTING B-TREES**
RT-MAC-9410, junho de 1994, 21 pp.

Renata Wassermann e Flávio S. Corrêa da Silva
TOWARDS EFFICIENT MODELLING OF DISTRIBUTED KNOWLEDGE USING EQUATIONAL AND ORDER-SORTED LOGIC
RT-MAC-9411, junho de 1994, 15 pp.

Jair M. Abe, Flávio S. Corrêa da Silva e Marcio Rillo
PARACONSISTENT LOGICS IN ARTIFICIAL INTELLIGENCE AND ROBOTICS.
RT-MAC-9412, junho de 1994, 14 pp.

Flávio S. Corrêa da Silva, Daniela V. Carbogim
A SYSTEM FOR REASONING WITH FUZZY PREDICATES
RT-MAC-9413, junho de 1994, 22 pp.

Flávio S. Corrêa da Silva, Jair M. Abe, Marcio Rillo
MODELING PARACONSISTENT KNOWLEDGE IN DISTRIBUTED SYSTEMS
RT-MAC-9414, julho de 1994, 12 pp.

Nami Kobayashi
THE CLOSURE UNDER DIVISION AND A CHARACTERIZATION OF THE RECOGNIZABLE Z-SUBSETS
RT-MAC-9415, julho de 1994, 29pp.

Flávio K. Miyazawa e Yoshiko Wakabayashi
AN ALGORITHM FOR THE THREE-DIMENSIONAL PACKING PROBLEM WITH ASYMPTOTIC PERFORMANCE ANALYSIS
RT-MAC-9416, novembro de 1994, 30 pp.

Thomaz I. Seidman e Carlos Humes Jr.
SOME KANBAN-CONTROLLED MANUFACTURING SYSTEMS: A FIRST STABILITY ANALYSIS
RT-MAC-9501, janeiro de 1995, 19 pp.

C.Humes Jr. and A.F.P.C. Humes
STABILIZATION IN FMS BY QUASI- PERIODIC POLICIES
RT-MAC-9502, março de 1995, 31 pp.

Fabio Kon e Arnaldo Mandel
SODA: A LEASE-BASED CONSISTENT DISTRIBUTED FILE SYSTEM
RT-MAC-9503, março de 1995, 18 pp.

Junior Barrera, Nina Sumiko Tomita, Flávio Soares C. Silva, Routo Terada
AUTOMATIC PROGRAMMING OF BINARY MORPHOLOGICAL MACHINES BY PAC LEARNING
RT-MAC-9504, abril de 1995,16 pp.

Flávio S. Corrêa da Silva e Fabio Kon
CATEGORIAL GRAMMAR AND HARMONIC ANALYSIS
RT-MAC-9505, junho de 1995,17 pp.

Henrique Mongelli e Routo Terada
ALGORITMOS PARALELOS PARA SOLUÇÃO DE SISTEMAS LINEARES
RT-MAC-9506, junho de 1995, 158 pp.

Kunio Okuda
PARALELIZAÇÃO DE LAÇOS UNIFORMES POR REDUÇÃO DE DEPENDÊNCIA
RT-MAC-9507, julho de 1995, 27 pp.

Valdemar W. Setzer e Lowell Monke
COMPUTERS IN EDUCATION: WHY, WHEN, HOW
RT-MAC-9508, julho de 1995, 21 pp.

Flávio S. Corrêa da Silva
REASONING WITH LOCAL AND GLOBAL INCONSISTENCIES
RT-MAC-9509, julho de 1995, 16 pp.

Julio M. Stern
MODELOS MATEMÁTICOS PARA FORMAÇÃO DE PORTFÓLIOS
RT-MAC-9510, julho de 1995, 43 pp.

Fernando Iazzetta e Fabio Kon
A DETAILED DESCRIPTION OF MAXANNEALING
RT-MAC-9511, agosto de 1995, 22 pp.

Flávio Keidi Miyazawa e Yoshiko Wakabayashi
POLYNOMIAL APPROXIMATION ALGORITHMS FOR THE ORTHOGONAL Z-ORIENTED 3-D PACKING PROBLEM
RT-MAC-9512, agosto de 1995, pp.

Junior Barrera e Guillermo Pablo Salas
SET OPERATIONS ON COLLECTIONS OF CLOSED INTERVALS AND THEIR APPLICATIONS TO THE AUTOMATIC PROGRAMMINIG OF MORPHOLOGICAL MACHINES
RT-MAC-9513, agosto de 1995, 84 pp.

Marco Dimas Gubitoso e Jörg Cordsen
PERFORMANCE CONSIDERATIONS IN VOTE FOR PEACE
RT-MAC-9514, novembro de 1995, 18pp.

Carlos Eduardo Ferreira e Yoshiko Wakabayashi
ANAIS DA I OFICINA NACIONAL EM PROBLEMAS COMBINATÓRIOS: TEORIA, ALGORITMOS E APLICAÇÕES
RT-MAC-9515, novembro de 1995, 45 pp.

Markus Endler and Anil D'Souza
SUPPORTING DISTRIBUTED APPLICATION MANAGEMENT IN SAMPA
RT-MAC-9516, novembro de 1995, 22 pp.

Junior Barrera, Routo Terada,
Flávio Corrêa da Silva and Nina Sumiko Tomita
*AUTOMATIC PROGRAMMING OF MMACH'S FOR OCR**
RT-MAC-9517, dezembro de 1995, 14 pp.

Junior Barrera, Guillermo Pablo Salas and Ronaldo Fumio Hashimoto
SET OPERATIONS ON CLOSED INTERVALS AND THEIR APPLICATIONS TO THE AUTOMATIC PROGRAMMING OF MMACH'S
RT-MAC-9518, dezembro de 1995, 14 pp.

Daniela V. Carbogim and Flávio S. Corrêa da Silva
FACTS, ANNOTATIONS, ARGUMENTS AND REASONING
RT-MAC-9601, janeiro de 1996, 22 pp.

Kunio Okuda
REDUÇÃO DE DEPENDÊNCIA PARCIAL E REDUÇÃO DE DEPENDÊNCIA GENERALIZADA
RT-MAC-9602, fevereiro de 1996, 20 pp.

Junior Barrera, Edward R. Dougherty and Nina Sumiko Tomita
AUTOMATIC PROGRAMMING OF BINARY MORPHOLOGICAL MACHINES BY DESIGN OF STATISTICALLY OPTIMAL OPERATORS IN THE CONTEXT OF COMPUTATIONAL LEARNING THEORY.
RT-MAC-9603, abril de 1996, 48 pp.