DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Relatório Técnico

RT-MAC-9405
Automatic Programming of
Binary Morphological Machines

Junior Barrera
Flávio S.C. da Silva
Gerald Jean Francis Banon

Abril 94

Automatic Programming of Binary Morphological Machines

Junior Barrera[1]
Flávio Soares Corrêa da Silva[2]
Gerald Jean Francis Banon[3]

[1, 2] Universidade de São Paulo, Departamento de Ciência da Computação
Cidade Universitária "Armando de Salles Oliveira", Caixa Postal 20570, 01452-990, São Paulo, SP, Brazil

[3] Instituto Nacional de Pesquisas Espaciais, Divisão de Processamento de Imagens
Av. dos Astronautas, 1752, Caixa Postal 515, 12201, São José dos Campos, SP, Brazil.

ABSTRACT

An important aspect of Mathematical Morphology is the description of set operators by a formal language, the Binary Morphological Language (BML), whose vocabulary are dilations, erosions, antidilations, antierosions, union and intersection. This language is complete (i.e. it can represent any set operator) and expressive (i.e. many useful operators can be represented as phrases with relatively few words). Since the sixties special machines, the Binary Morphological Machines (BMMach's), have been built to implement the BML with increasing efficiency. However, designing useful BMMach programs is not an elementary task. Recently, much research effort has been addressed to automating the programming of BMMach's. The goal is to find suitable knowledge representation formalisms to describe operations over geometric structures and to translate them into BMMach programs. We propose an approach for the automated programming of translation invariant operators: operators are described either by logical expressions or by sample input-output lists and translated into BMMach programs by semantic evaluation, probably approximately correct (PAC) learning or automated deduction over abstract operations. The generated operators are optimized by transformations on their decomposition structure. A priori knowledge is modeled by associating probability distributions to occurrences of images. The design of optimal and suboptimal morphological filters can be seen as particular cases of the proposed approach. Some examples illustrate the main ideas presented.

1. INTRODUCTION

Binary Image Analysis is an important tool for various areas, such as industrial process control, office automation, quantitative microscopy, etc.

A natural model of a procedure for Binary Image Analysis is a *set operator* (i.e. a mapping over a powerset). *Mathematical Morphology* (MM) is a general framework to study operators over complete lattices [1], which includes set operators [2 3]. The central paradigm of MM is the decomposition of operators in terms of four classes of *elementary operators*: dilations, erosions, antidilations and antierosions.

The rules for the representation of set operators in terms of the elementary operators can be described as a formal language [4], the *Binary Morphological Language* (BML). The vocabulary of the BML are the four classes of elementary operators and the operations of union and intersection. A phrase of the BML is called a *morphological operator*. The BML is complete (i.e. it can represent any set operator) and expressive (i.e. many useful operators can be represented as phrases with relatively few words). Moreover, some morphological operators can be simplified into equivalent operators that use a smaller number of elementary operators. An implementation of this language is called a *Binary Morphological Machine* (BMMach), and a *program of a BMMach* is an implementation of a morphological operator on this machine.

Nowadays, there are many commercially available BMMach's implemented in hardware [5 6 7 8] or emulated in software, [9 10 11] which have been intensively used for Binary Image Analysis [3].

Programming a BMMach can be a very difficult task. In order to help the non experts in MM to use BMMach's, some tools have been proposed to automate the design of programs. These tools act as translators of the user knowledge about the problem, expressed as high level abstract procedures, into morphological operators.

The existing systems are based on two main paradigms: *rule-based expert systems* [12 13 14] and *automated deduction*. [15] Expert systems employ a combination of stored heuristics to generate a set operator, while automated theorem provers give a constructive proof that there exists an operator equivalent to a logical description. The steps in this constructive proof can be interpreted directly as applications of the morphological operators. Each morphological operator generated by the expert system or the theorem prover can be incorporated to the system, respectively, as a new rule or lemma. So these systems can learn and improve their efficacy with use.

The main goal of this paper is to present a general model, based on formal approaches, to automate the design of programs for BMMach's. The proposed model is composed by the following main steps: description of a set operator as a logical expression or as an input-output list of sample images; translation of this description into a morphological operator by semantic evaluation, PAC learning or automated deduction; simplification of the derived morphological operator using contextual information or transformation of decomposition structures. For sake of simplicity, we stay restricted to the automatic programming of translation invariant (t.i.) operators.

An important characteristics of the proposed model is the symbiosis of three formal techniques of Artificial Intelligence: automated deduction, PAC learning and program transformation via algebraic rewrite rules. A fundamental fact exploited for this integration is the Canonical Decomposition Theorem for t.i. operators [16].

In section 2, we give some basic definitions and results from MM theory. In section 3, we present the PAC learning model. In section 4, we present our proposed model for automatic programming of BMMach's. In section 5, we show an application example. Finally, in section 6, we present some further discussion.

2. MATHEMATICAL MORPHOLOGY

For the automatic programming of BMMach's some relevant aspects of the theory of MM on sets are: the canonical decompositions of set operators, the optimal filter design and the transformation of decomposition structures.

2.1 Canonical decompositions

Let $\mathcal{P}(E)$ be the collection of all subsets of a finite non empty subset E (i.e. the collection of all binary images). Let $\subset$ be the usual inclusion relation on sets. Let X^c be the complementary set of a subset X of E. We know that $(\mathcal{P}(E), \subset)$ is a complete Boolean lattice. [17] The intersection and union of X_1 and $X_2 \in \mathcal{P}(E)$ are, respectively, $X_1 \cap X_2$ and $X_1 \cup X_2$.

The set E is assumed to be an Abelian group with respect to a binary operation denoted by +. The zero element of $(E, +)$ is denoted by o. Let X^t be the *transpose* of a subset X, that is, $X^t = \{y \in E : y = -x, x \in X\}$. A set X is said to be *symmetric* if $X^t = X$.

For any $h \in E$ and $X \subset E$, the set $X + h = \{y \in E : y = x + h, x \in X\}$ is called the *translate* of X by h. In particular, $X_o = X$.

A *set operator* is any mapping defined from $\mathcal{P}(E)$ into itself. A set operator ψ is called *translation invariant* (t.i.) if and only if (iff)

$$\psi(X + h) = \psi(X) + h \quad (X \in \mathcal{P}(E),\ h \in E).$$

The *kernel* $\mathcal{K}(\psi)$ of a t.i. set operator ψ is the subcollection of $\mathcal{P}(E)$ defined by

$$\mathcal{K}(\psi) = \{X \in \mathcal{P}(E) : o \in \psi(X)\}.$$

DEFINITION 2.1 Let $B \in \mathcal{P}(E)$. The t.i. set operators δ_B and ε_B defined by

$$\delta_B(X) = \{x \in E : (B^t + x) \cap X \neq \emptyset\} \quad (X \in \mathcal{P}(E))$$

and

$$\varepsilon_B(X) = \{x \in E : (B + x) \subset X\} \quad (X \in \mathcal{P}(E))$$

are called, respectively, *dilation* and *erosion* by B. □

The t.i. set operators δ^c_B and ε^c_B (i.e. the composition of the complement operator, respectively, with dilation and erosion) are called, respectively, *antidilation* and *antierosion*, and are denoted δ^*_B and ε^*_B. The parameter B that characterizes a dilation or an erosion is called a *structuring element*.

A subcollection $[A, B]$ of $\mathcal{P}(E)$, with $A \subset B$, is called a *closed interval* iff

$$[A, B] = \{X \in \mathcal{P}(E) : A \subset X \subset B\}.$$

The sets A and B are called, respectively, the *left* and *right extremities* of the closed interval.

A useful property of erosions and antidilations is that they are sufficient to decompose any t.i. operator in standard forms. Banon and Barrera stated such property as the following theorem: [16]

THEOREM 2.1 (*Canonical Decomposition Theorem*) Let ψ be a t.i. operator and $\mathcal{K}(\psi)$ be its kernel, then

$$\psi(X) = \bigcup \{\varepsilon_A(X) \cap \delta^*_{B^c}(X) : [A, B] \subset \mathcal{K}(\psi)\} \quad (X \in \mathcal{P}(E)).$$ □

This representation theorem may lead to inefficient computational representations for most t.i. operators, in the sense that a smaller family of erosions and antidilations may be sufficient to represent the same operator.

A closed interval contained in a subcollection $\mathcal{U}$ of $\mathcal{P}(E)$ is called *maximal in* $\mathcal{U}$ if no other interval contained in $\mathcal{U}$ properly contains it. The set $\boldsymbol{B}(\psi)$ of all the maximal closed intervals contained in $\mathcal{K}(\psi)$ is called the *basis* of ψ.

Banon and Barrera also proved that $\mathcal{K}(\psi)$ can be replaced by $\boldsymbol{B}(\psi)$ in the decomposition formula, [16] that is,

$$\psi(X) = \bigcup \{\varepsilon_A(X) \cap \delta^*_{B^c}(X) : [A, B] \in \boldsymbol{B}(\psi)\} \quad (X \in \mathcal{P}(E)).$$

In practice, the interesting operators are the ones that depend on a local neighborhood. A t.i. operator is called *locally defined within a window* $W \subset E$ iff

$$h \in \psi(X) \Leftrightarrow h \in \psi(X \cap (W + h)),$$

for all $h \in E$ and $X \in \mathcal{P}(E)$.

If ψ is a locally defined t.i. operator within the window W and $[A, B] \in \boldsymbol{B}(\psi)$, then A, $B^c \in \mathcal{P}(W)$. In other words, the structuring elements which characterize the erosions and the antidilations used in the decomposition of locally defined operators are subsets of the window W.

An important property of t.i. operators is that they are closely related to Boolean functions. Let $\{0, 1\}^{\mathcal{P}(W)}$ denote the set of Boolean functions defined from $\mathcal{P}(W)$ to $\{0, 1\}$. For each Boolean function $b \in \{0, 1\}^{\mathcal{P}(W)}$ we can associate the locally defined t.i. operator ψ_b, given by

$$\psi_b(X) = \{x \in E : b((X - x) \cap W) = 1\} \quad (X \in \mathcal{P}(E)).$$

Conversely, for each locally defined t.i. operator ψ we can associate the Boolean function $b_\psi \in \{0, 1\}^{\mathcal{P}(W)}$ given by

$$b_\psi(X) = 1 \Leftrightarrow o \in \psi(X) \quad (X \in \mathcal{P}(W)).$$

The mappings $b \mapsto \psi_b$ and $\psi \mapsto b_\psi$ are reciprocal functions.

In the standard representation of b by a sum of minterms, each Boolean variable is associated with a point in W and each minterm corresponds to an element of the kernel of ψ_b. The usual simplification of Boolean functions of the switching theory [18] can be applied to simplify the representation of ψ_b. The determination of the so-called *prime implicants* of the Boolean function b by the Quine-McCluskey method leads exactly to the basis of ψ_b, since the prime implicants of b correspond to the maximal closed intervals contained in $\mathcal{K}(\psi)$. In a prime implicant, the Boolean variables not complemented and complemented define, respectively, the left extremity and the complement of the right extremity of the closed interval.

2.2 Optimal filter design

As shown by Dougherty [19], the Canonical Decomposition Theorem can be useful to design optimal morphological filters. To do so, we must put the morphological operators in the context of a theory of estimation: morphological operators are seen as statistical estimators that are a function of a random variable.

Let the mapping $\boldsymbol{X}$ with values in $\mathcal{P}(E)$ be a *random set* with probability distribution $p(\boldsymbol{X})$. Let η be a set operator over $\mathcal{P}(E)$ which models the noise that corrupts the images. Let ψ be the set operator over $\mathcal{P}(E)$ which estimates the thrue image.

Let $|Z|$ denote the cardinality of a subset Z of E. The *mean absolute error* (MAE) committed by the estimator ψ, when estimating the random set $\boldsymbol{X}$ from the random set $\eta(\boldsymbol{X})$, denoted $MAE(\psi)$, is given by

$$MAE(\psi) = \sum |(\boldsymbol{X} \cap \psi(\eta(\boldsymbol{X}))^c) \cup (\psi(\eta(\boldsymbol{X})) \cap \boldsymbol{X}^c)|.p(\boldsymbol{X}, \psi(\eta(\boldsymbol{X}))),$$

where $p(\boldsymbol{X}, \psi(\eta(\boldsymbol{X})))$ is the joint probability distribution of the random sets $\boldsymbol{X}$ and $\psi(\eta(\boldsymbol{X}))$.

The estimator operator ψ is called an *optimal estimator* iff $MAE(\psi) \leq MAE(\beta)$, for any estimator operator β over $\mathcal{P}(E)$.

Assuming that the estimator ψ is a locally defined t.i. operator within a window W with cardinality n, under stationary condition on $\boldsymbol{X}$, the MAE expression simplifies to

$$MAE(\psi) = \sum |\boldsymbol{x}_{n+1} - f_\psi(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)|.p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n+1}),$$

where $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ are the n Boolean random variables observed in the window W, $\boldsymbol{x}_{n+1}$ is the Boolean random variable to be estimated and $p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n+1})$ is their joint probability distribution.

Hence, reducing the search of optimal estimators to the class of locally defined t.i. operators implies in the search of Boolean functions, that (as discussed in section 2.1) can be translated into canonical decompositions of locally defined t.i. set operators. Clearly, the distribution $p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$ is determined by the distribution $p(\boldsymbol{X})$.

2.3 Transformation of decomposition structures

A set operator may be represented by an infinite number of BML phrases that are *synonyms* (i.e. different phrases which express the same operator). When implementing a set operator in a BMMach, we are interested in cheap realizations for the operator, that is, BML phrases which involve the smallest possible number of elementary operators.

There exists a bijection between the set of the t.i. operators and the set of subcollections $\mathcal{P}(\mathcal{P}(E))$. In other words, each subcollection in $\mathcal{P}(\mathcal{P}(E))$ is the kernel of a unique t.i. operator. As, in the finite case, there exists a bijection between the set of collections of maximal intervals of the subcollections in $\mathcal{P}(\mathcal{P}(E))$ and $\mathcal{P}(\mathcal{P}(E))$ itself, then there exists a bijection between the set of basis of t.i. operators and the set of t.i. operators itself.

Thus, for each operation on the set of t.i. operators corresponds an equivalent one on the set of t.i. operator basis and conversely. As the BML can be reduced to compositions of unions, dilations and complementations (or, equivalently, intersections, erosions and complementations), it suffices to understand the composition of these operators with an arbitrary t.i. operator to be able to construct *transformations of decomposition structures* of t.i. morphological operators.

Let $\mathfrak{U}$ be a collection of closed interval in $\mathcal{P}(\mathcal{P}(E))$. Let $\mathrm{Max}(\mathfrak{U})$ denotes the collection of maximal closed intervals of $\cup\mathfrak{U}$.

PROPERTY 2.1 Let ψ and ψ' be two arbitrary t.i. set operators. The following equalities hold:

1) $\boldsymbol{B}(\delta_B\psi) = \mathrm{Max}(\{[X + b, Y + b] : [X, Y] \in \boldsymbol{B}(\psi), b \in B'\})$;

2) $\boldsymbol{B}(\psi^c) = \mathrm{Max}(\{[A, B] \in \mathcal{P}(\mathcal{P}(E)) : \forall [X, Y] \in \boldsymbol{B}(\psi),\ |A \cap Y^c| = 1 \text{ and } |B^c \cap X| = 1\})$;

3) $\boldsymbol{B}(\psi \vee \psi') = \mathrm{Max}(\boldsymbol{B}(\psi) \cup \boldsymbol{B}(\psi'))$.

The Example 2.1 illustrates how to apply Property 2.1 in order to compute the basis of any t.i. set operator from a synonym phrase in the BML.

EXAMPLE 2.1 Let the operator ψ be defined by $\psi = \iota \wedge \varepsilon_B^c$, where ι denotes the identity operator (i.e. $\iota(X) = X,\ \forall X \in \mathcal{P}(E)$) and B be a symmetric subset which contains the origin o. This operator can be represented in terms of unions, dilations and complementations as $\psi = (\iota^c \vee \delta_B{}^c\iota^c)^c$. To compute the basis of ψ, we compute *incrementally* the basis of the operator compositions from the basis of the identity operator:

1) $\boldsymbol{B}(\iota) = \{[\{o\}, E]\}$; 2) $\boldsymbol{B}(\iota^c) = \{[\emptyset, \{o\}^c]\}$; 3) $\boldsymbol{B}(\delta_B\iota^c) = \{[\emptyset, \{-b\}^c] : b \in B\}$;

4) $\boldsymbol{B}(\delta_B{}^c\iota^c) = \{[B, E]\}$; 5) $\boldsymbol{B}(\iota^c \vee \delta_B{}^c\iota^c) = \{[B, E], [\emptyset, \{o\}^c]\}$;

6) $\boldsymbol{B}(\psi) = \{[\{o\}, \{b\}^c] : b \in B \cap \{o\}^c\}$.

The rules of Property 2.1 could also be used to solve the inverse problem, that is, to go from the canonical decomposition to simpler decomposition structures. However, this is a much more complex problem, since for each state of the algorithm (analogously to the transition of states in a chess game) there are many possible next states Jones [20] studied this problem in the case of increasing t.i. operators (i.e. t.i. operators ψ such that $X \subset Y \Rightarrow \psi(X) \subset \psi(Y), \forall X, Y \in \mathcal{P}(E)$).

3. PAC LEARNING

Computational Learning Theory [21, 22] is one of the first attempts to construct a mathematical model for a cognitive process. It provides a framework for studying a variety of algorithmic processes, such as those currently in use for training artificial neural networks. Here, we apply this framework to design set operators.

We understand *concept* as a subset of objects in a predefined domain. An *example* of a concept is an object from the domain together with a label indicating whether the object belongs to the concept. If the object belongs to the concept, it is a *positive example*, otherwise it is a *negative example*. *Concept learning* is the process in which a *learner* constructs a good approximation to an unknown concept, given a small number of *examples* and some prior information on the concept to be learned. In the following, we formalize these ideas.

Let Σ be a set, called the *alphabet* to describe examples. In this paper, Σ will be the *Boolean alphabet* $\{0, 1\}$. We denote the set of n-tuples of elements of Σ by Σ^n. Let X be a subset of Σ^n. We define a *concept*, over the alphabet Σ, as a function $c : X \to \{0, 1\}$.

The set X will be referred to as the *example space*, and its members as *examples*. An example $y \in X$ for which $c(y) = 1$ is known as a *positive example*, and an example for which $c(y) = 0$ is known as a *negative example*. So, provided that the domain is known, c determines, and is determined by, its set of positive examples. So sometimes it is helpful to think of a concept as a set in that way.

The set of all possible concepts to be learned will be referred to as the *hypothesis space* and denoted by H. The concept $t \in H$ to be determined is called the *target concept*. The problem is to find a concept $h \in H$, called the *hypothesis*, which is a good approximation for t.

A *sample of length* m is just a sequence of m examples, that is, an m-tuple $\boldsymbol{x} = \{x_1, x_2, \ldots, x_m\}$ in X^m. The sequence may contain the same value more than once. A *training sample* $\boldsymbol{s}$ is an element of $X^m \times \{0,1\}$, that is,

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

where the x_i are examples and the b_i are 0 or 1. The value of b_i is given by a *teacher* and specifies whether x_i is a positive or a negative example. There are no contradictory labels, so that if $x_i = x_j$ then $b_i = b_j$.

A *learning algorithm* is simply a function L which assigns to any training sample $\boldsymbol{s}$ for a target concept $t \in H$ a hypothesis $h \in H$. We write $h = L(\boldsymbol{s})$.

Let μ be a *probability distribution* (or *probability measure*) on X. Given a target concept $t \in H$, we define the *error of any hypothesis* $h \in H$, with respect to t, as the probability of the event $h(x) \neq t(x)$, that is, .

$$\boldsymbol{er}_{\mu}(h,t) = \mu\{x \in X : h(x) \neq t(x)\}.$$

When a given set X is provided with the structure of a probability space, the product set X^m inherits this structure from X. The corresponding distribution on X^m is denoted μ^m. Usually, the components of the m-tuple $(x_1, x_2, \ldots, x_m)$ are assumed to be "independent" variables, each distributed according to the probability distribution μ on X.

Let $S(m,t)$ denote the set of training samples of length m for a given target concept t, where the examples are drawn from an example space X. As there is a bijection $\phi : X^m \to S(m,t)$ for which $\phi(\boldsymbol{x}) = \boldsymbol{s}$, the following equality hold

$$\mu^m\{\boldsymbol{s} \in S(m,t) : \boldsymbol{s} \text{ has property } P\} = \mu^m\{\boldsymbol{x} \in X^m : \phi(\boldsymbol{x}) \in S(m,t) \text{ has property } P\}.$$

DEFINITION 3.1 We say that the algorithm L is a *probably approximately correct* (PAC) learning algorithm for the hypothesis space H if, given two real numbers ε and δ $(0 < \varepsilon, \delta < 1)$, then there is a positive integer $m_0 = m_0(\varepsilon, \delta)$ such that for any target concept $t \in H$, and for any distribution μ on X, whenever $m \geq m_0$,

$$\mu^m\{\boldsymbol{s} \in S(m,t) : \boldsymbol{er}_{\mu}(L(\boldsymbol{s})) < \varepsilon\} > 1 - \delta.$$

The function $m_0 = m_0(\varepsilon, \delta)$ is called *example complexity*. □

A learning algorithm L for H *is consistent* iff, given any training sample $\boldsymbol{s}$ for a target concept $t \in H$, the output hypothesis agrees with t on the examples in $\boldsymbol{s}$, that is, $h(x_i) = t(x_i)$ $(1 \leq i \leq m)$. For a given $\boldsymbol{s} \in S(m,t)$, we denote by $H[\boldsymbol{s}]$ the set of all hypotheses consistent with $\boldsymbol{s}$, that is,

$$H[\boldsymbol{s}] = \{h \in H : h(x_i) = t(x_i)\ (1 \leq i \leq m)\}.$$

Given $\varepsilon \in (0,1)$, the set

$$B_{\varepsilon} = \{h \in H : \boldsymbol{er}_{\mu}(h) \geq \varepsilon\}$$

is called the set of ε-*bad hypothesis for* t.

We say that the hypothesis space H is *potentially learnable* if, given two real numbers ε and δ $(0 < \varepsilon, \delta < 1)$, there is a positive integer $m_0 = m_0(\varepsilon, \delta)$ such that, whenever $m \geq m_0$

$$\mu^m\{\boldsymbol{s} \in S(m,t) : H[\boldsymbol{s}] \cap B_{\varepsilon} = \emptyset\} > 1 - \delta$$

for any probability distribution μ on X and any $t \in H$.

The following theorems are proved to hold: [22]

THEOREM 3.1 If H is potentially learnable, and L is a consistent learning algorithm for H, then L is PAC.

THEOREM 3.2 Any finite hypothesis space is potentially learnable.

The following example presents an algorithm for learning concepts which are Boolean functions.

EXAMPLE 3.1 Let $D_{n,k}$ be the space of all those Boolean functions of n variables which can be expressed as the disjunction of monomials of length at most k ($n \geq k > 1$). The following learning algorithm was proposed by Valiant: [21]

set h := disjunction of all monomials of length at most k.
for i:=1 to m do
  if $b_i = 0$ and $h(x_i) = 1$
    then delete monomials $\mathfrak{M}$ for which $\mathfrak{M}(x_i) = 1$;
$L(s) := h$

The hypothesis space $D_{n,k}$ has cardinality bound above by $2^{(2n)^k}$ (i.e. $|D_{n,k}| < 2^{(2n)^k}$). Thus, as Valiant's algorithm is consistent, it is a PAC learning algorithm.

According to Anthony and Biggs, [22] the example complexity of Valiant's algorithm is

$$m_0(\varepsilon,\delta) = \lceil (k/\varepsilon) \ln 2n + (1/\varepsilon) \ln (1/\delta) \rceil ,$$

for any distribution μ and with no restriction in the hypothesis space $H = D_{n,k}$.

4. A MODEL FOR AUTOMATIC PROGRAMMING OF BMMach's

The proposed model for the *automatic programming of BMMach's* is presented as a data flow diagram in Figure 1. The goal of this model is to translate the *user knowledge* about the *target operator* and the *application domain* into *morphological operators* or, equivalently, into *programs for BMMach's*.

The user knowledge can be represented by two distinct knowledge representation formalisms: *logical expressions* or *input-output lists of examples*. Logical expressions are used alternatively in two distinct representations forms: *full specification of operators* and *abstract image operations*.

In full specification of operators, *windowed images* (i.e. $X - h \cap W, h \in E$) and image operators are seen as propositional formulae in which basic propositions are associated to points in a finite discrete square W of size $(2n)^2$, centered at the origin o. In order to make the correspondence between images and propositions more natural, we index the propositions with the position of the corresponding points in W. The alphabet of this language consist of a finite set of basic propositions $\Phi = \{q_{0,0}, p_{-n,-n}, \ldots, p_{n,n}\}$ and the conventional connectives (i.e. $\rightarrow$, $\neg$, $\vee$ and $\wedge$). Proposition $p_{i,j}$ is associated to the point (i, j) of an windowed image (i.e. $(X - h) \cap W$), while proposition $q_{0,0}$ is associated to a point of the output image (i.e. $\psi(X) \cap \{h\}$). Basic propositions correspond to *black* and *white* points, respectively, when they are *true* and *false*. Finally, a set operator is represented as a formula of the form $P \leftrightarrow q_{0,0}$, in which P contains only propositions $p_{i,j}$ (e.g. see Example 4.2).

In abstract image operations, as presented in Joo [15], morphological operators are structured in "packets" that characterize abstract operations frequently employed in Binary Image Analysis (e.g. difference between subsets, or some special operators like sieve filter) and geometrical properties like convexity, topological structure, or size. The abstract operations and properties are encoded as first-order sentences, whose semantics is given by the packed of morphological operators they represent (e.g. assuming subsets to be represented by the constant symbols $a, b, c, \ldots$ and variables $X, Y, Z, \ldots$, *difference* can be represented as a function *diff*(X, Y), whose interpretation can be the subset $X \cap Y^c$). This particular interpretation forces (in the model theory

sense) some relations between the resulting functions and predicates. From those, some relations are selected and encoded as either axioms or inference rules in a corresponding first-order theory of MM, necessarily correct but not necessarily (and most probably not) complete.

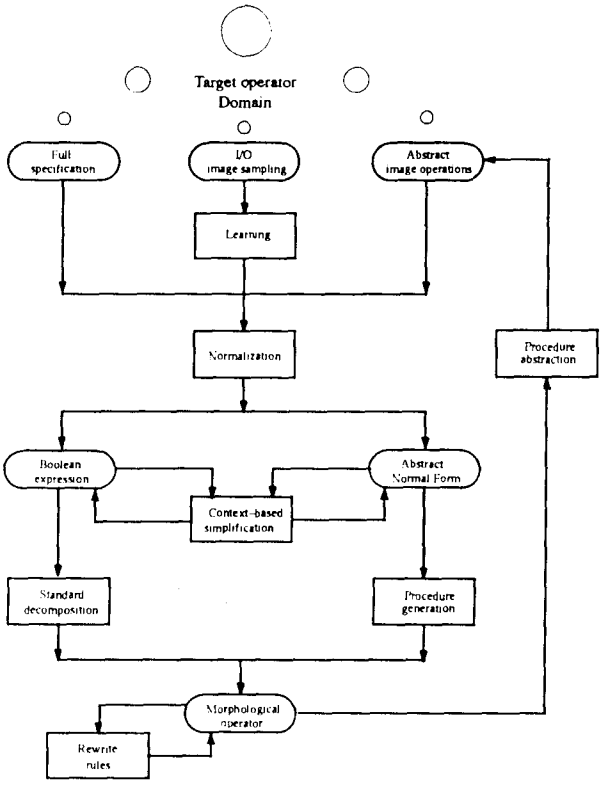


Fig. 1 – A model for the automatic programming of BMMach's

When the user does not have complete information to specify a first-order function or relation to describe an operator or it is to complex to do that, he can specify approximately this operator by a list of input-output examples. These examples are pairs consisting of an input windowed image (i.e. $(X - h) \cap W$) and the corresponding value in the output image (i.e. $\psi(X) \cap \{h\}$). This list of examples feeds a learning algorithm, similar to the one presented in Example 3.1. The output of this learning algorithm is a Boolean function which characterizes a set operator that is consistent with the examples and satisfy the PAC quality ε, δ.

The following example illustrates the use of the learning algorithm of Example 3.1 to obtain a locally defined t.i. set operator.

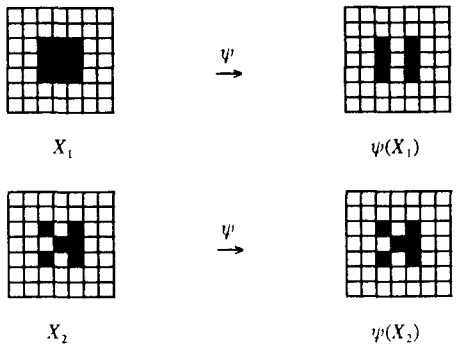
EXAMPLE 4.1 Let ψ be a set operator locally defined within the horizontal window $W = \{(-1,0),(0,0),(1,0)\}$, defined by the input-output pairs of images of Figure 2a. The corresponding Boolean function f_ψ, defined in Figure 2b, is

$$f_\psi(x_1, x_2, x_3) = (\neg x_1 \wedge x_2) \vee (x_2 \wedge \neg x_3),$$

where x_1 is associated to the point $(-1,0)$, ... The canonical form of the corresponding set operator is

$$\psi = (\varepsilon_{\{o\}} \wedge \delta^*_{\{(-1,0)\}}) \vee (\varepsilon_{\{o\}} \wedge \delta^*_{\{(1,0)\}}).$$

Note that this set operator, which performs the extraction of vertical edges, is fully specified by the input-output image pairs $(X_1, \psi(X_1))$ and $(X_2, \psi(X_2))$. If only the pair $(X_1, \psi(X_1))$ were given the learned function would be f_ψ □



a

x_1 x_2 x_3	$f_\psi(x_1\ x_2\ x_3)$	$f_\psi(x_1\ x_2\ x_3)$
0 0 0	0	0
0 0 1	0	0
0 1 0	1	1
1 0 0	0	0
0 1 1	1	1
1 1 0	1	1
1 0 1	0	1
1 1 1	0	0

b

Fig. 2 – a) Input–output image pairs. b) Truth table of the Boolean function.

Once created, the full specification of operators and the abstract image operations are normalized. The Boolean functions generated by the learning algorithm also are normalized into disjunctive normal forms. The normalization process build standard inputs for the following processes and eliminate redundant logical expressions. A full specification of an operator can be translate by semantic evaluation into a Boolean function and conversely.

General descriptions in the form of normalized logical expressions or normalized Boolean functions may be simplified by context, which may be given either by further logical expressions or by a list of examples (i.e. a list of input images). By context simplification we understand the appropriate use of available a priori information about the image domain in order to simplify the specification of the set operators. When a context is introduced (i.e. the domain of set operators is restricted) a large number of operators become equivalent, and we can choose an operator between simpler ones (i.e. ones corresponding to shorter expressions). When the context is specified by a list of examples the logical description of the set operator acts as the ***teacher*** in the learning process. When the context and the operator are specified by logical expressions symbolic simplification can be used.

The following example illustrate the context simplification, when both operator and context are given in the full specification form.

EXAMPLE 4.2 Let's study the problem of recognizing 3×3 squares within a 5×5 window, positioned arbitrarily in the image. The full specification for this shape recognition task is

$$\bigwedge_{-1 \le i,j \le 1} p_{ij} \wedge \bigwedge_{-2 \le i \le 2} \neg p_{i2} \wedge \bigwedge_{-2 \le i \le 2} \neg p_{i-2} \wedge \bigwedge_{-1 \le j \le 1} \neg p_{2j} \wedge \bigwedge_{-1 \le i \le 1} \neg p_{-2j} \leftrightarrow q_{00}$$

The corresponding basis of the set operator ψ is given by the single interval $[A,B]$, where $A = \{-1,0,1\}^2$ and $B^c = \{-2,-1,0,1,2\}^2 \cap A^c$. Now assume that we know a priori that the only occurring shapes in our input images are 3×3 squares and 3×3 squares with a hole in the middle. In other words, we know that whenever we find a 3x3 cross, the cross must belong to the target shape. The context simplification gives (see [23] for details)

$$\bigwedge_{i \in \{-1,1\}} p_{i,j} \quad \wedge \quad \bigwedge_{j \in \{-1,1\}} p_{i,j} \quad \wedge \quad p_{0,0} \leftrightarrow q_{0,0}$$

The basis of the simplified operator ψ' reduce to the single interval $[A', B']$, where $A' = \{(-1,0),(1,0),(0,0),(0,-1),(0,1)\}$ and $B' = E$. Observe that $\psi' = \varepsilon_{A'}$ is simpler than $\psi = \varepsilon_A \wedge \delta^*_{B^c}$, since its decomposition involves just one erosion, while ψ decomposition involves an erosion and an antidilation. □

The generation of morphological operators from Boolean functions is straightforward (see section 2.1), while the translation of abstract logical descriptions into morphological operators can be done from constructive proofs of relations between properties of input and output images represented as first-order theorems. By using some automatic theorem proving techniques (like the Resolution-based techniques adopted by Joo [15], or even more sophisticated techniques like proof planning [24]), given the specification of a set operator task in terms of expect relations between properties of input/output images, we can obtain an automatically generated morphological operator that performs the desired transformation.

The structure of the generated morphological operators could be changed by using program transformation techniques which explore Property 2.1.

This model should improve with use, since the set operators generated are incorporated to the model as new logical expressions.

5. APPLICATION EXAMPLE

Here, we illustrate the use of PAC learning in order to generate a simple set operator which performs *optimal images restoration*. We take the image of Figure 3a and corrupt it with subtractive punctual noise, with an unknown distribution (Figure 3b). We choose arbitrarily the 3×3 cross as the window, i.e. $W = \{(0,1),(-1,0),(0,0),(0,1),(0,-1)\}$. From the original image and the corrupted one, we build a frequency table which estimate the distribution of $\mathcal{P}(W) \times \{0,1\}$. The elements of $\mathcal{P}(W)$ are taken from the noise image $\eta(X)$ by the operation $(\eta(X) - h) \cap W$, and the corresponding ideal restoration values (i.e. 0 or 1) are taken from the original image X by the rule (1 iff $\{o\} \cap (X - h) \neq \emptyset$). In order to establish a training sample, for each subset in $\mathcal{P}(W)$ is chosen the most frequent value associate (i.e. 0 or 1). Figure 3c shows the frequency table generated from the images of Figures 3a and 3b. This training sample feeds the PAC learning algorithm of Example 3.1.

For this problem, the example complexity, with $\varepsilon = \delta = 0.01$, is $m_0 = 1,611$. The number of examples experimentally used was 3,844 (i.e. 62^2). The learned Boolean function is

$$f_\psi(x_1, x_2, x_3, x_4, x_5) = x_1 \vee x_3 \vee x_5,$$

where x_1 is associated to the point $(0,1)$, x_2 is associated to the point $(-1,0)$, ... The corresponding set operator is

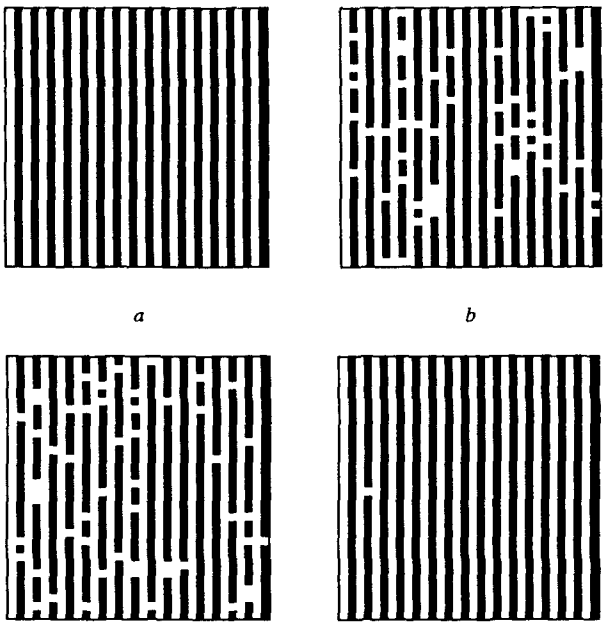
$$\psi' = \varepsilon_{\{(0,1)\}} \vee \varepsilon_{\{(0,0)\}} \vee \varepsilon_{\{(0,-1)\}}.$$

Observe that in the PAC learning of the optimal restoration operator a third quality parameter could be defined, besides ε and δ. This parameter should reflect the correctness of the statistical decision done in order to establish the training sample. Figure 3d presents the image of Figure 3a corrupted by other realization of the same noise source. Figure 3e presents the result of the restoration by the generated operator ψ'.

6. CONCLUSION

We introduced the use of PAC learning theory in MM, by deriving the canonical decomposition of t.i. set operators through learning of Boolean functions. The proposed methodology was applied to the problem of designing optimal t.i. morphological filters.

We defined the rules for the transformation of decomposition structures for the general case of t.i. set operators and showed how they can be used to compute the basis of any t.i. set operator for which is known a representation as a BML phrase.



a *b*

d *e*

x_1 x_2 x_3 x_4 x_5	$f_v(x_1 \ldots x_5)$	Frequency
0 0 0 0 0	0	18
~~0 0 0 0 0~~	~~1~~	~~6~~
0 0 0 0 1	1	17
0 0 0 1 0	0	175
0 0 1 0 0	1	15
0 0 1 0 1	1	156
0 1 0 0 0	0	175
0 1 0 1 0	0	1554
1 0 0 0 0	1	18
1 0 0 0 1	1	152
1 0 1 0 0	1	155
1 0 1 0 1	1	1403

c

Fig. 3 – Image restoration example. a) Thruth image. b) Training noisy image. c) Table of frequency. d) Test noisy image. e) Restored image from the test noisy image.

We presented a general model for the automatic programming of BMMach's, based on formal approaches. The proposed model is supported conceptually by results on set operator decompositions, PAC learning and applied automated deduction, and integrate logical descriptions with sample based descriptions.

The advantage of the generation of set operators via automated theorem proving is its ergonomicity, since image transformations and operators are presented in terms of intuitively sound concepts encoded as "packets" of morphological operators. Thus, being reliable, easy to check and generally efficient. Moreover, the generated procedures can still be easily translated as propositional expressions and further normalized, producing final implementations equally efficient to the ones resulting from Boolean specification or PAC learning.

The main drawback of the theorem proving approach when compared to the full specification or learning approaches is its inherent incompleteness: the first-order theory of "packets" of morphological operators is not guaranteed to capture every t.i. set operator, even if we encode every possible relation between "packets" of operations.

This last observation characterizes the complementarity of the two approaches, and makes clear why we believe it is important that they coexist.

7. ACKNOWLEDGMENTS

The authors thank Professors Routo Terada and Ricardo Bianconi for interesting discussions concerning PAC learning, and the research student Nina Sumiko Tomita, for the implementation of programs used in the application example.

The authors have received partial financial support from CNPq (Conselho Nacional de Desenvolvimento Cientifico e Tecnológico) and FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo), grants 91/3532-2 and 93/0603-01.

8. REFERENCES

1. J. Serra, *Image Analysis and Mathematical Morphology. Volume 2: Theoretical Advances*, Academic Press, London, 1988.

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

3. J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, New York, 1982.

4. J. Barrera and G. Banon, Expressiveness of the morphological language, *Image Algebra and Morphological Image Processing III*, pp. 264-275, SPIE, San Diego, California, July 1992.

5. J. C. Klein and J. Serra, "The texture analyser," *Journal of Microscopy*, vol. 95, pt. 2, pp. 343-356, Apr. 1972.

6. P. Husson, J. P. Derutin, P. Bonton and J. Gallice, "Elementary processor with mathematical morphology implemented in VLSI and intented for systolic architecture," *Signal Processing IV: Theories and Aplication*, EURASIP, pp. 1561-1564, 1988.

7. J. C. Klein and R. Peyhard, "PIMM1, An image processing ASIC based on mathematical morphology," *IEEE's A ASIC Seminar and Exhibit*, Rochester, New York, pp. 25-28, Sep. 1989.

8. K. S. Huang, B. K. Jenkins and A. A. Sawchuk, "Binary image algebra and optical cellular logic processor design," *Computer Vision, Graphics and Image Processing*, vol. 45, pp. 295-345, 1989.

9. B. Läy, "Description des programmes du logiciel morpholog," *CGMM Rapport interne N-904*, Fontainebleau, 1984.

10. J. Barrera; G. Banon; R. Lotufo A Mathematical Morphology Toolbox for the KHOROS System, *Image algebra and Morphological Image Processing V*, SPIE, San Diego, California, July 1994.

11. C. Gratin, "Élaboration d'un logiciel de morphologie mathématique sur micro-ordinateur," *CMM Rapport interne*, Fontainebleau, 1989.

12. M. Schimitt Mathematical Morphology and Artificial Intelligence: an automatic programming system. *Signal Processing*. Signal Processing, v.16, n.4, pp. 389-401, Apr. 1989.

13. R. Vogt *Automatic Generation of morphological set recognition algorithms*, Spring-Verlag Inc., New York, 1989.

14. H. Zhao *Systeme Expert et Morphologie Mathématique*, PhD thesis, École des Mines de Paris, 1992.

15. H. Joo *Automatic Morphology*, PhD thesis, University of Washington, 1991.

16. G. J. F. Banon and J. Barrera, "Minimal representations for translation invariant set mappings by mathematical morphology," vol. 51, no. 6, *SIAM Journal of Applied Mathematics*, pp. 1782-1798, Dec. 1991.

17. G. Birkhoff, *Lattice Theory*, American Mathematical Society, Providence, Rhode Island, 1967.

18. F. Hill and G. Peterson *Introduction to switching theory and logical design*, John Wiley & Sons, New York, Second Edition, 1974.

19. E. Dougherty Optimal Mean-square n-observation Digital Morphological Filters. *CVGIP: Image Undestanding*, v. 55, pp. 36-54, 1992.

20. R. Jones Transformations of the Basis Representation into Cascade Representations, *International Workshop on Mathematical Morphology and its applications to Signal Processing*, 1993.

21. L. Valiant A Theory of the Learnable, *Comm. ACM*, v. 27, pp. 1134-1142, 1984.

22. M. Anthony and N. Biggs *Computacional Learning Theory. An introduction*, Cambridge University Press, 1992.

23. F. Côrrea da Silva and J. Barrera *Automating the Generation of Procedures to Analyse Binary Images*, Relatório Técnico do IME-USP, RT-MAC-9402, Jan. 1994.

24. S. Negrete Planes de Prueba y Demonstracion Automatica de Teoremas con Sugerencias, *Proceedings, X Simpósio Brasileiro de Inteligência Artificial*, pp. 93-110, Brazil, 1993.

RELATÓRIOS TÉCNICOS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Instituto de Matemática e Estatística da USP

A listagem contendo os relatórios técnicos anteriores a 1992 poderá ser consultada ou solicitada à Secretaria do Departamento, pessoalmente, por carta ou e-mail(mac@ime.usp.br).

J.Z. Gonçalves, Arnaldo Mandel
COMMUTATIVITY THEOREMS FOR DIVISION RINGS AND DOMAINS
RT-MAC-9201, Janeiro 1992, 12 pgs

J. Sakarovitch
THE "LAST" DECISION PROBLEM FOR RATIONAL TRACE LANGUAGES
RT-MAC 9202, Abril 1992, 20 pgs

Valdemar W. Setzer, Fábio Henrique Carvalheiro
ALGOTRITMOS E SUA ANÁLISE (UMA INTRODUÇÃO DIDÁTICA)
RT-MAC 9203, Agosto 1992, 19 pgs

Claudio Santos Pinhanez
UM SIMULADOR DE SUBSUMPTION ARCHITECTURES
RT-MAC-9204, Outubro 1992, 18 pgs

Julio M. Stern
REGIONALIZAÇÃO DA MATRIZ PARA O ESTADO DE SÃO PAULO
RT-MAC-9205, Julho 1992, 14 pgs

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

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

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

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 pgs

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 pgs

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

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 pgs

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 pgs

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

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 pgs

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

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 pgs