# Image operator learning and applications

Igor S. Montagner, Nina S. T. Hirata, Roberto Hirata Jr.
Institute of Mathematics and Statistics
University of São Paulo,
São Paulo, Brazil
igordsm,nina,hirata@ime.usp.br

***Abstract*—High-level understanding of image contents has been receiving much attention in the last decade. Low level processing figures as a building block in this framework and it also continues to play an important role in several specific tasks such as in image filtering and colorization, medical imaging, and document image processing. The design of image operators for these tasks is usually done manually by exploiting characteristics specific to the domain of application. An alternative design approach is to use machine learning techniques to estimate the transformations. Given pairs of images consisting of a typical input and respective desired output, the goal is to estimate an operator that transforms the inputs into the desired outputs. In this tutorial we present a rigorous mathematical formulation to the framework of learning locally defined and translation invariant transformations, practical procedures and strategies to address typical machine learning related issues, application examples, and current challenges. We also include information about the code used to generate the application examples.**

***Keywords*-Image Operator Learning; Machine Learning; Image Processing**

## I. Introduction

Computer Vision research has witnessed huge advances in the last decades. Its application scope has been largely extended, with the type and contents of processed images shifting from the ones obtained under strictly controlled conditions to those freely obtained without or with little control (for instance, images or videos obtained with mobile device cameras), and from application specific images to natural scene images. Nowadays there are complex vision systems that are capable of recognizing specific objects such as faces [1], or a few dozen of objects [2], in natural or uncontrolled images. There are also initiatives that integrate vision systems into autonomous cars [3], surveillance systems, and so on.

These modern applications require, in general, high level understanding of image contents which includes tasks such as object detection and recognition [4]. The most recent advances in image processing methods rely on the use of a vast set of image features combined with machine learning techniques. We can cite SIFT [5] as a successful example of these methods. SIFT combines a clever way of extracting image descriptors that are invariant under many common deformations and variations (illumination, scale, rotation) and algorithms for matching the extracted descriptors. They can be used to detect objects as well as to classify images. Another successful example is the convolutional neural network [6], [7] which has been already proven useful in many application problems (http://deeplearning.net/demos/).

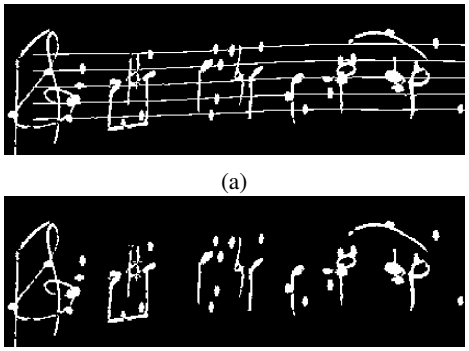
Despite the trend towards higher level processing, low level processing still figures as an important task in many problems that involve image analysis. Some examples where lower level processing, including pixel level processing, are important and required are in document image analysis, medical image analysis, and segmentation of object contours for diverse applications. Besides that, low level features are also used as building blocks for higher level approaches [8], [4].

Low level processing relies on local features of the images and thus local image operators are suitable for these tasks. For instance, filtering was a trendy topic just a few decades ago and many useful filters such as Wiener filter, median filter, and morphological filters have been proposed. Morphological operators, which have emerged around the same period, is based on locally probing geometrical structures present in images and constitute a powerful tool for image processing [9], [10].

A large class of local image operators can be modeled as morphological operators. These operators are formally studied in the field of mathematical morphology, where images are modeled as elements of a complete lattice (a partially ordered set with the meet and join defined for each pair of elements) [9], [11]. For example, binary images are modeled as subsets of the image domain, and binary image operators as set transformations. Even if we constrain the set of morphological operators to those that are translation invariant and locally defined with respect to a neighborhood window $W$, called $W$-operators, we still have a very large set of operators.

Designing a solution based on these type of image operators requires a considerable degree of knowledge and experience from the users. An alternative way of designing them is by using machine learning techniques. In order to be useful, such approach should rely preferably only on high level input from the users. In that sense, many previous approaches for local image operator learning have proposed the use of input-output pairs of images as the source of training data [12], [13], such as the ones shown in Figure 1. This type of approach has been successfully applied on filtering tasks in the decade of 80 and 90. Median and stack filter design and template matching filters are examples of such applications [13], [12], [14].

In addition, several machine learning topics such as feature learning, dictionary learning [15], kernel methods [16], classifier combination [17] have also emerged and established themselves as important tools in machine learning applica-



(a)

(b)

Figure 1: Input-output pair for the staff removal task.

tions. Many of these techniques have been motivated by image processing and computer vision related applications.

The goal of this tutorial paper is to present a comprehensive view on image operator learning and, in particular, of $W$-operators. The content of the paper is organized as follows. In Section II we present the fundamentals of image operators, some important properties and results regarding their representation and local characterization. In Section III we review the notion of optimal operators and describe their estimation procedure from training images, inserting the image operator design problem into the framework of machine learning. Specifically we show the connections between learning image operators and classifier training. Then, in Section IV we present some strategies used to address machine learning related issues that pose challenges in the classifier design process. These strategies explore two-level design approach, operator space constraining, kernel approximations and other ideas. In Section VI we show examples of successful applications. In Section VII we present some concluding remarks.

## II. Basic concepts on image operators

We consider that digital images are defined on the discrete grid[1] $\mathbb{E} = \mathbb{Z}^2$. Thus images are represented by functions of the form $f : \mathbb{E} \to K$, where $K = \{0, 1, \ldots, k-1\}$ denotes the set of gray levels of the image. Typical 8-bit gray-scale images use $k = 256$ and binary images use $k = 2$. The value of $f$ at a given point $p \in \mathbb{E}$ corresponds to the gray level of image $f$ at that point. The set of all images defined on $\mathbb{E}$ with gray-levels in $K$ is denoted as $K^{\mathbb{E}}$, i.e., $K^{\mathbb{E}} = \{f \mid f : \mathbb{E} \to K\}$.

An image operator is any mapping of the form $\Psi : K^{\mathbb{E}} \to K^{\mathbb{E}}$. Given an image $f$, $[\Psi(f)](p)$ denotes the gray level of the transformed image $\Psi(f)$ at point $p$. Note that an operator may be such that its input is a gray-scale image and its output is a binary image. Since $\{0, 1\} \subset \{0, 1, \ldots, 255\}$, for convenience we use gray level set $K$ both for the input and output images, unless an explicit distinction is required.

[1]In practice, images have a finite support (usually a rectangular region). However, from a theoretical point of view it is convenient to deal with domains closed under translation. Thus to avoid the additional definitions needed to add such property to a finite domain, in this section we consider $\mathbb{E} = \mathbb{Z}^2$.

### A. Translation invariance and local definition

Two important properties of image operators are *translation invariance* and *local definition*. First let us define the translation of an image. Given an image $f : \mathbb{E} \to K$, its translation by $q \in \mathbb{E}$ is denoted $f_q$ and defined for every $p \in \mathbb{E}$ as $f_q(p) = f(p-q)$, where the symbol $-$ denotes the usual vector subtraction operation. An operator $\Psi$ is said to be translation-invariant if $[\Psi(f)]_q = \Psi(f_q)$.

Local definition refers to the property that states that the value of the transformed image $\Psi(f)$ at any point $p$ depends only on the image values on a limited neighborhood of $p$. Formally, an operator is a window operator if there exists a finite set $W \subseteq \mathbb{E}$ such that $[\Psi(f)](p) = [\Psi(f|_{W'_p})](p)$ for every $p \in \mathbb{E}, f \in K^{\mathbb{E}}$, and $W' \supseteq W$ [9]. Here $f|_{W'_p}$ is an image with support $W'_p$, with the same value of $f$ on each point within $W'_p$ and with value 0 on each point out of $W'_p$.

Operators that are translation-invariant and locally defined are known as $W$-operators. An important consequence of these two properties is the characterization of these operators by a local function [9]. Any $W$-operator $\Psi$ can be characterized, for any point $p$, by a local function $\psi : K^W \to K$ as follows:

$$[\Psi(f)](p) = \psi(f_{-p}|_W)\,. \tag{1}$$

The above equation means that the output value of the transformed image $\Psi(f)$ of an input image $f$, at point $p$, can be computed by a local function $\psi$ applied on the subimage $f_{-p}|_W$, which is $f|_{W_p}$ (the image $f$ restricted to $W_p$) shifted by $-p$ (thus $f_{-p}|_W$ is an image defined on $W$). We consider that $W$ is defined around the origin $o$. Figure 2 shows a $3 \times 3$ window (with its center at the origin) and its translation $W_p$.

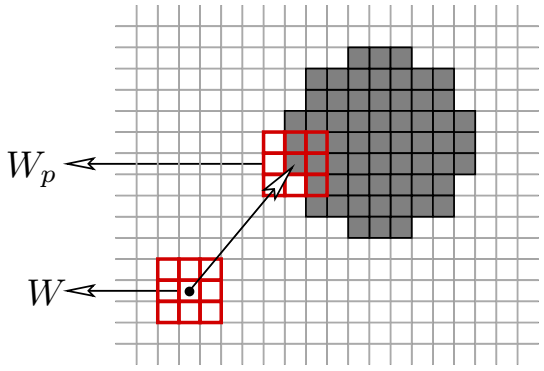



Figure 2: A window $W$, its translation $W_p$, and $f|_{W_p}$.

Operators as the ones defined above are extensively studied in the field of Mathematical Morphology, within the framework of lattice theory [9], [18]. Images are modeled as elements of an algebraic structure known as lattice (partially ordered sets with the join and meet operations defined for each pair of elements). The set of image operators is itself a lattice and thus algebraic properties of these operators are also studied with the help of lattice-theoretical concepts.

An important theoretical result regarding translation-invariant lattice operators is their canonical representation,

uniquely characterized by a set known as kernel, as a supremum of elementary operators [19]. The representation for the class of binary image operators is detailed next.

### B. Binary operators

A binary image $f : \mathbb{E} \to \{0, 1\}$ can be equivalently represented as a subset $S_f \subseteq \mathbb{E}$ through the bijection $p \in S_f \iff f(p) = 1$. Thus, the set of all possible binary images can be represented by the power set of $\mathbb{E}$, $\mathcal{P}(\mathbb{E}) = \{S \,|\, S \subseteq \mathbb{E}\}$.

In the following, we detail the representation theorem for translation invariant binary image operators. Let us introduce some additional notations and definitions. The translation of a set $S$ (corresponding here to an image translation) by a vector $q$ is denoted $S_q$ and defined as $S_q = \{p + q \,|\, p \in S\}$. Its transpose is defined as $\check{S} = \{-p \,|\, p \in S\}$ and its complement as $S^c = \{p \,|\, p \notin S\}$. The set $[A, B] = \{S \in \mathcal{P}(\mathbb{E}) \,|\, A \subseteq S \subseteq B\}$ is called an interval with extremities $A$ and $B$ and it can be understood as a set of images.

Two elementary operators, characterized by a structuring element $B \subseteq \mathbb{E}$ and defined based on set operations, are:

- *Erosion:* $\varepsilon_B(S) = \{p \in E \,|\, B_p \subseteq S\}$
- *Dilation:* $\delta_B(S) = \{p \in E \,|\, \check{B}_p \cap S \neq \emptyset\}$

A basic operator, called sup-generating [20], defined as $\Lambda_{(A,B)}(S) = \{p \in E \,|\, A_p \subseteq S \subseteq B_p\}$ can be decomposed in terms of these elementary operators:

$$\begin{aligned} \Lambda_{(A,B)}(S) &= \{p \in E \,|\, A_p \subseteq S \subseteq B_p\} \\ &= \{p \in E \,|\, A_p \subseteq S\} \cap \{p \in E \,|\, S \subseteq B_p\} \\ &= \varepsilon_A(S) \cap \{p \in E \,|\, (B_p)^c \subseteq S^c\} \\ &= \varepsilon_A(S) \cap \varepsilon_{B^c}(S^c)\,. \end{aligned} \tag{2}$$

Sup-generating operators are also known as wedge operators [9] and are closely related to the hit-or-miss operators [11].

The canonical representation theorem for translation-invariant set operators was first presented in [20]. It extended the decomposition results previously presented by G. Matheron [21] for the class of increasing operators (that is, the operators that preserve the partial order relation of the lattice). The decomposition structure of a set operator $\Psi$ relies on the kernel of the operator, which is defined as

$$\mathcal{K}(\Psi) = \{S \in \mathcal{P}(E) \,|\, o \in \Psi(S)\} \tag{3}$$

where $o$ denotes the origin of $\mathbb{E}$.

Note that the kernel of the sup-generating operator parameterized by an interval $[A, B]$ is the interval $[A, B]$ itself, that is,

$$\begin{aligned} \mathcal{K}(\Lambda_{(A,B)}) &= \{S \,|\, o \in \Lambda_{(A,B)}(S)\} \\ &= \{S \,|\, o \in \{A_p \subseteq S \subseteq B_p\}\} \\ &= \{S \,|\, A_o \subseteq S \subseteq B_o\} \\ &= \{S \,|\, A \subseteq S \subseteq B\} = [A, B]\,. \end{aligned} \tag{4}$$

Thus, they are also called as *interval operators*.

Having the above definitions, we are ready to state the representation theorem.

**Theorem** (Sup-representation): Let $\Psi : \mathcal{P}(\mathbb{E}) \to \mathcal{P}(\mathbb{E})$ be a translation-invariant set operator and let $\mathcal{K}(\Psi)$ be its kernel. Then, the following result holds.

$$\Psi(S) = \bigcup \{\Lambda_{(A,B)}(S) : [A, B] \subseteq \mathcal{K}(\Psi)\}\,. \tag{5}$$

A proof can be found in [9]. Since operators are uniquely characterized by kernels, we have a canonical representation of the operators as a union of a set of interval operators. The above decomposition has redundant terms (for instance, $[A_1, B_1] \subseteq [A_2, B_2]$ implies $\Lambda_{(A_1,B_1)}(S) \subseteq \Lambda_{(A_2,B_2)}(S)$). Removing the redundant term leads to an equivalent canonical minimal representation. The reader may refer to [20] for more details.

Some algebraic properties of image operators define specific subclasses of operators. Among them, we cite increasingness (those that preserve the order relation, i.e., those such that $S_1 \subseteq S_2 \Longrightarrow \Psi(S_1) \subseteq \Psi(S_2)$), and anti-extensiveness (those such that $\Psi(S) \subseteq S$).

If $\Psi$ is a $W$-operator, then the kernel elements can be constrained to subsets of $W$ and the interval operators can be seen as detectors of a collection of templates contained in $W$[20], [9]. By assigning a binary logical variable to each point in $W$, the local function of $\Psi$ can be understood as a logical function. In the following we present a few examples to illustrate these concepts.

### C. Examples

*1) Erosion and dilation:* Let $B = \{(-1, 0), (0, 0), (1, 0)\}$ be a structuring element, centered at the origin $o = (0, 0)$. According to the definition, $p \in \varepsilon_B(S) \iff B_p \subseteq S$. Thus, $\varepsilon_B$ is a $W$-operator, with $W = B$. Considering three logic variables $x_1, x_2, x_3$ associated to the points in $W$ and letting $x_1 = 1 \iff (-1, 0) \in B \cap S_{-p}$ and similarly for $x_2$ and $x_3$ with respect to $(0, 0)$ and $(1, 0)$, we have $p \in \varepsilon_B(S) \iff x_1 = 1$, $x_2 = 1$ and $x_3 = 1$. Hence, the local function of $\varepsilon_B$ is $\psi_\varepsilon(x_1, x_2, x_3) = x_1\, x_2\, x_3$.

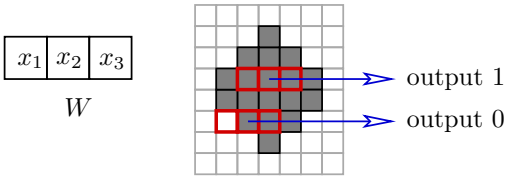



Figure 3: Local function $\psi_\varepsilon$ applied on two distinct points. If $W_p$ is entirely contained in the foreground, then output is 1 at $p$; otherwise, it is 0.

Analogously, for the dilation operator we have $p \in \delta_B(S) \iff \psi_\delta(x_1, x_2, x_3) = x_1 + x_2 + x_3 = 1$. Note that the logic product term $x_1\, x_2\, x_3$ in $\psi_\varepsilon(x_1, x_2, x_3)$ corresponds to the interval $[B, B]$, and the terms $x_1$, $x_2$ and $x_3$ in $\psi_\delta$ correspond respectively to $[\{(-1, 0)\}, B]$, $[\{(0, 0)\}, B]$ and $[\{(1, 0)\}, B]$. More specifically, each of the product terms in the logic function corresponds to an interval operator. There is a one to one correspondence between the sum of products form

of logic functions and the representation of image operators as supremum of interval operators.

*2) Internal contour points:* Let a point in $S$ be a contour point if it has at least one of its 4 neighbors in the background. Such contour detector operator can be defined using a 5-point cross window. Figure 4 shows the eight patterns within the 5-point cross window such that the central point corresponds to a contour point with a background point above it. Letting $x_3$ be the logic variable associated to the central point of $W$, and the other four variables as shown in Fig. 4, the logic function that defines the operator is $\psi(x_1, x_2, x_3, x_4, x_5) = \overline{x}_1\, x_3 + \overline{x}_2\, x_3 + \overline{x}_4\, x_3 + \overline{x}_5\, x_3$. Each product term corresponds to an interval operator. Figure 4 illustrates the elements in the interval $[\{x_3\}, \{x_2, x_3, x_4, x_5\}]$.

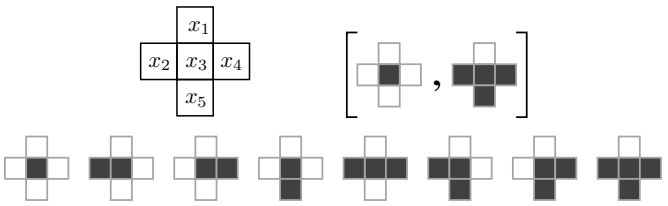



Figure 4: (Top left) the logic variables assigned to each point in the cross window. (Top right) between brackets, the extremities of the interval corresponding to the product term $\overline{x}_1\, x_3$. (Bottom) the eight patterns included in the interval and that corresponds to the product terms $\overline{x}_1\, \overline{x}_2\, x_3\, \overline{x}_4\, \overline{x}_5$, $\overline{x}_1\, x_2\, x_3\, \overline{x}_4\, \overline{x}_5$, $\overline{x}_1\, \overline{x}_2\, x_3\, x_4\, \overline{x}_5$, and so on. Filled squares correspond to foreground pixels and non-filled ones to the background pixels.

### *D. Extension to gray-scale operators and some properties*

Operators for gray-scale images can also be framed in the lattice theoretical formulation by modeling gray-scale images as elements of a complete lattice. An initially proposed extension was based on the notion of umbra [22], [11], [9], and later on the representation of images as functions. A natural partial order relation between two gray level functions $f, g \in K^{\mathbb{E}}$ is defined as:

$$f \leq g \Longleftrightarrow f(p) \leq g(p), p \in \mathbb{E}\,. \tag{6}$$

Regarding this extension, a main difference between binary and gray-scale is the complement operator. While the notion of complement is clearly defined for sets (binary images), there is no natural definition of the complement operation for gray-scale images, and thus some variants have been proposed [9], [23], [10]. Regardless of the specific definition of the complement operation, the decomposition as a supremum of interval operators also holds for gray-scale image operators. Since the sup-representation of gray-scale image operators may involve an extremely large number of basic operators, their explicit representation does not favor computation. Alternative representations of the local function are more adequate for computational processing, as will become clear afterwards. Therefore here we do not include a detailed description of representation related results. For more details, the reader may refer to [19], [9], [24], [25], [26].

## III. Formulation of the learning problem

A common procedure to solve image processing tasks is by means of combinations of multiple basic operators. Such combination may include sequential pipelines, decision and iteration subprocesses. Finding a satisfactory solution requires a trial and error approach in which several configurations of the composition, varying the basic operators and their parameters, must be assembled and tested [8], [27], [10].

Since these are time consuming tasks, many efforts to automate part or the whole design process have been proposed in the field. Among them, the first approaches for designing non linear window operators emerged in the 1980s. They were, however, restricted to specific subclasses of filters such as the median and stack filters [28], [14].

The representation of $W$-operators as a supremum of interval operators, and in particular, its local characterization by a function (see Eq. 1) opened a new and general path for learning image operators from training images [29], [24].

In order to model the learning process, a performance measure to be optimized must be defined. An image $f$ is assumed to be a realization of a random process $\mathbf{F}$ and input-output pairs of images $(f, g)$, representing observed images and respective desired transformations, are assumed to be realizations of a pair of random processes $(\mathbf{F}, \mathbf{G})$, with a joint distribution $P(\mathbf{F}, \mathbf{G})$. Under these assumptions, the learning problem is formulated based on a statistical approach. The goal is to find an image operator $\Psi$ such that $\Psi(\mathbf{F})$ is the best possible approximation of $\mathbf{G}$.

The notion of proximity between $\mathbf{G}$ and $\Psi(\mathbf{F})$ can be modeled by means of a probabilistic measure. Let $\Psi$ be a $W$-operator characterized by the local function $\psi$ defined on $W$. Then, for any specific point $p \in \mathbb{E}$, the *mean absolute error* (MAE) and the *mean square error* (MSE) are defined as:

$$MAE\langle\Psi\rangle = E[\,|\psi(\mathbf{f}_{-p}|_W) - \mathbf{g}(p)|\,]\,, \tag{7}$$

$$MSE\langle\Psi\rangle = E[\,(\psi(\mathbf{f}_{-p}|_W) - \mathbf{g}(p))^2\,]\,. \tag{8}$$

Considering wide sense stationarity and local definition, position $p$ is irrelevant and thus both the MAE and the MSE can be written in terms of a local process $(\mathbf{X}, \mathbf{y})$, where $\mathbf{X}$ is a random process that models observations on input images through $W$ and $\mathbf{y}$ is a random variable associated to the central point of $W$ on the output image:

$$MAE\langle\Psi\rangle = E[\,|\psi(\mathbf{f}_{-p}|_W) - \mathbf{g}(p)|\,] = E[\,|\psi(\mathbf{X}) - \mathbf{y}|\,]\,. \tag{9}$$

The MAE and the MSE are minimized when the estimator $\psi(X)$ is, respectively, the mode and the mean of observed outputs $(y)$ for each observation $X$. In the binary setting, both errors are equivalent and the optimal operator is the one characterized by the following local function:

$$\psi(X) = \begin{cases} 1, & \text{if } P(1|X) > P(0|X), \\ 0, & \text{if } P(0|X) > P(1|X), \\ 0 \text{ or } 1 & \text{if } P(0|X) = P(1|X) = 0.5 \end{cases} \tag{10}$$

This can be easily seen by expanding the MAE equation:

$$\begin{aligned} MAE\langle\Psi\rangle &= E[\,|\psi(\mathbf{X}) - \mathbf{y}|\,] \\ &= \sum_{(X,y)} P(X,y)|\psi(X) - y| \\ &= \sum_{(X,y)} P(X)P(y|X)|\psi(X) - y| \\ &= \sum_{X} P(X)\Big[P(0|X)\psi(X) + P(1|X)(1-\psi(X))\Big] \end{aligned}$$

Note that the observations $X$ such that $\psi(X) = 1$ correspond to the kernel elements.

### A. Estimating from training images

Local functions that characterize the optimal operators can therefore be estimated from training images. For the binary case, it is the function given in Eq. 10. Note that each training image $f$ has a finite support, which we denote $D_f$. Given an input-output pair $(f, g)$ of training images, with $D_f = D_g$, samples are collected at each point $p$ such that $W_p \subseteq D_f$ by sliding $W$ over $f$. The local sample at $p$ is the pair $(f_{-p}|_W, g(p))$. All collected observations are pooled together. The observed image patch, $f_{-p}|_W$, is called a *window image*.

However, several possible patterns usually will not be observed in the training images and thus the estimated function will be only partially defined. Logic minimization algorithms and decision trees have been used previously for the generalization step of binary [29], [30] and gray-scale [31] operators, respectively.

After the generalization of the function, the resulting operator can be applied on test images similar to those used for the estimation. Performance of the operators are usually measured in terms of its empirical MAE/MSE on test images. The empirical MAE of an operator with respect to a pair $(f, g)$ is computed as the pixel-wise mean absolute difference between $\Psi(f)$ and $g$ :

$$Err = \frac{1}{|D'_f|} \sum_{p \in D'_f} \Big|\psi(f_{-p}|_W) - g(p)\Big| \qquad (11)$$

where $D'_f$ denotes set of points in $D_f$ at which samples are collected.

### B. Connections with Machine Learning

Under the perspective of machine learning, the local function $\psi$ to be estimated can be seen as a classifier that assigns a gray level value in $K$ for each pattern in $W$. Thus, the image operator learning problem described above can be modeled as a typical classifier learning problem. Just to recall, in classifier learning the goal is to learn a target function $h : \mathcal{X} \to \mathcal{Y}$, from a set of samples $\mathcal{S} \subseteq \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the feature space representing the observations and $\mathcal{Y}$ is the set representing the class labels. Hereafter we adopt some notations that are common in the machine learning field. They will be introduced below as we point out the connections between image operator learning and classifier learning.

The step of sliding a window $W$, $|W| = m$, over the training images and collecting samples of window images can be seen as a process of transforming raw data (image pairs) into a collection of feature vectors $x_i \in \mathcal{X} = K^m$ and respective labels $y_i \in \mathcal{Y} = K$. Therefore, this corresponds to a *feature extraction* step. An illustration of this step is shown in Figure 5. As it can be seen, window constrained subimages extracted from the input images are flattened (in raster order) to generate the feature vectors. For a window image $x_i$ extracted at a pixel $p$, the class label $y_i$ is the value of the desired output image at the same pixel.

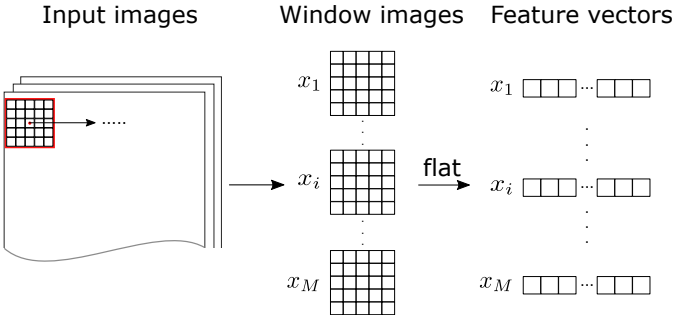



Figure 5: Illustration of the feature extraction process (output images not shown).

Estimation of the optimal local function corresponds to the classifier training step. Although the lattice theoretical model provides a canonical representation for these functions, it may consist of a very large number of intervals and it may not be suitable from a computational point of view. Thus, one can consider learning the local function without fixing its representation structure. The task of completing the function definition (generalization) can be left to machine learning algorithms.

By dividing the training procedure into two steps, feature extraction and classifier training, we can build very flexible $W$-operator learning systems. A typical image operator training procedure consists of the following steps:

1) separate input-output pairs of images into training and test images
2) extract feature vectors and respective labels from the training images to build the training set.
   Each set of identical window images can be reduced to one exemplar with its class label being the estimate value that minimizes the error measure (Eq. 10 if images are binary). This reduces the amount of training data. Moreover, if the classifier assigns consistent label for each feature vector in the reduced training set, then an optimal empirical error on the full training set will be achieved. However, the impact on generalization is not clear.
   One can also consider a set of general features extracted from each window image rather than just the flattened window image. Although this does not improve training error, it may greatly affect generalization performance. Two very simple examples of this type of feature

processing are quantization (reducing the number of gray levels in an image) and normalization. Other more complicated representations may be computed from the images. Notice that as long as only information from the window images is used, no theoretical results on $W$-operators are affected.

3) train a classifier from the training set (part of the training set can be used for validation).
   We can also change classifiers to match the characteristics of the input patterns or to enforce a specific property on the resulting operator. For instance, when dealing with normalized data, suitable learning models can be used.
4) take the trained classifier as the desired local function and apply it on the test images, generating $\Psi(f)$
5) compare, for each test pair $(f_j, g_j)$, the resulting image $\Psi(f_j)$ with the expected output $g_j$ and compute the relevant metrics (empirical MAE/MSE, accuracy, precision, recall, F-measure, etc)
   Note that, for binary output, the MAE is equivalent to the $0-1$ loss and, therefore, minimization of the MAE is equivalent to maximization of the accuracy.

The training/testing procedure of $W$-operators is schematized in Fig. 6.

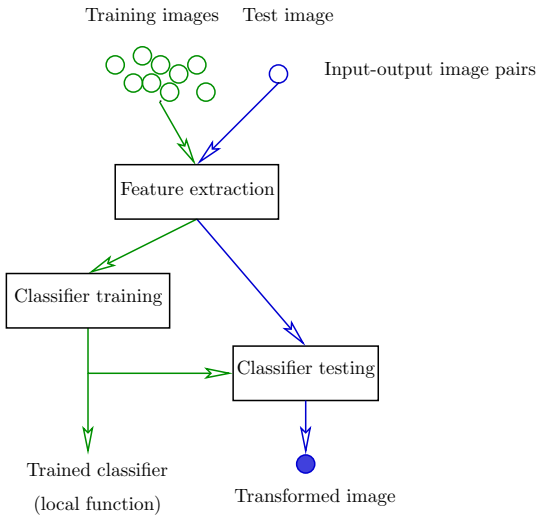



Figure 6: A typical $W$-operator training/testing process. The flow at the left (in green) represents the training process and the one at the right (in blue) represents the testing process. Multiple images pairs can be used for training (note that each circle represents an input-output pair).

## IV. Strategies to mitigate learning related issues

As discussed above, after a training set is built by extracting the feature vectors from input-output pairs of images, any methods available for classifier training can be applied. As in any usual machine learning problem, learning in the context of image operator design is also subject to several computational and statistical issues.

A typical question that often arises refers to how many training examples are necessary to obtain a good approximation of the desired image operator. Practice shows that it largely depends on the complexity of image content and processing task, as well as image resolution. These factors influence the choice of the window; the window should be large enough to discriminate relevant patterns. If we consider $k$ gray levels, the number of different possible patterns for a window of size $m$ is $k^m$, a quantity that may be beyond available computing resources. Moreover, since the number of available training images is usually limited, large windows tend to generate poor empirical errors, due to generalization error. Therefore, one of the challenging issues is to determine an appropriate window that is simultaneously useful for discrimination and that does not result in large generalization error.

Several approaches can be used to address these challenges. For instance, constraining the class of operators reduces the classifier hypothesis space, improving variance error. Examples of constrained classes are the aperture [32] and stack [33] filters. A two-level training strategy [30] is also an approach that has been proposed to mitigate the difficulties.

### A. Aperture operators

To overcome the need to deal with an extremely large number of distinct window images when grayscale images are considered, a subclass called *aperture* operators has been proposed in [34], [31]. The class consists of operators that are locally defined in the gray-level domain in a similar way $W$-operators are locally defined in the spatial domain.

Let us first define gray-level translation. Let $f : \mathbb{E} \to K$ be a grayscale image and $c \in K$. The grayscale *translation*[2] of $f$ by $c$ is given, for any $p \in \mathbb{E}$, by $(f + c)(p) = f(p) + c$.

The local definition in the gray-level domain is based on a grayscale (range) window $R = \{-r, \ldots, r\}$, where $r$ is a positive integer. The *windowing* of $f$ at level $v$ by the grayscale window $R$ is the function $f|R_v$ given by $(f|R_v)(p) = \min\{\max\{-r, f(p) - v\}, r\}$. That is, the range window is translated vertically such that its center matches $v$. Then, $f|R_v$ is such that the difference $f(p) - v$ is constrained to $R$.

A characteristic function $\psi : W \to K$ is called *locally defined* in $R$ iff, for any $g \in K^W$, we have $\psi(g) = g(o) + \beta_{g(o)}(g|R_{g(o)})$, where $\beta_y$, for any $y \in K$, is a function from $W$ to $R$. Thus, an *aperture operator* is a $W$-operator that is characterized by a function $\psi$ that is locally defined in $R$. That is, for any $f : \mathbb{E} \to K$, there is $\beta_\psi$ such that $[\psi(f)](p) = f(p) + \beta_\psi([f_{-p}|_W]|R_{f(p)})$. Instead of performing the windowing of $f$ at level $f(p)$, one can consider other levels such as the mean or median value of $f|_{W_p}$. In practical terms, in a similar way we can control the size of the spatial window $W$, we can control the height (aperture) of the range window $R$.

[2]Let us for now ignore the fact that $f(p)$ may surpass the maximum value, $k - 1$.

### B. Two level approach

The two-level approach [30] is based on combining several image operators through a second stage of training, as shown in Figure 7. First, $n$ operators are trained individually, each one with respect to a window $W_i$. Input images $f$ are then processed with these operators yielding $n$ transformed images $\Psi_i(f)$, $i = 1, 2, \ldots, n$. A new feature vector, called *second level pattern*, to be used in the second training stage is then built by concatenating the pixel values $[\Psi_i(f)](p)$, $i = 1, 2, \ldots, n$, for each $p \in D'_{\Psi_i(f)}$. Second level patterns have, thus, $n$ feature components. In this scheme, the resulting

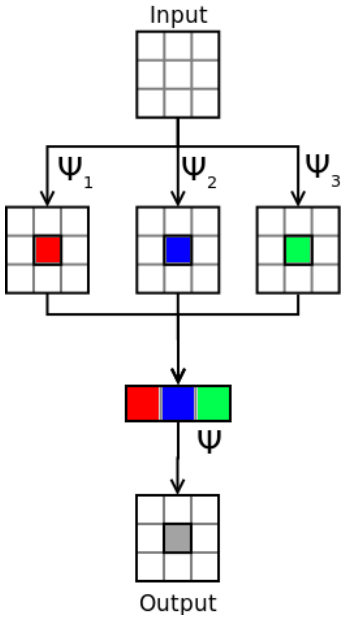



Figure 7: Two-level training scheme. Values of the three processed images, respectively by $\Psi_1$, $\Psi_2$ and $\Psi_3$, at a same point are concatenated to form a feature vector, which is then classified by the second-level operator $\Psi$.

final operator is called a two-level operator, with the individual operators that are combined, $\Psi_i(f)$, $i = 1, 2, \ldots, n$, being called first level operators, and the one that combines, $\Psi$, being called the second level operator (or combiner).

The composition of $W$-operators is also a $W$-operator. If the composition is sequential, the local function that characterizes the resulting $W$-operator is usually related to a larger support window [30]. This strategy can, therefore, be understood as being similar to classifier combination [17], and in particular to that cases in which multiple classifiers are designed using distinct subsets of features and then their outcomes are combined to reach a final classification. This strategy may be useful to avoid generalization errors that are due to high dimensionality.

In this two-level framework, deciding the number of operators to be combined and choosing windows for each of them are the two main challenges. On the other hand, the framework enables the application of clever machine learning techniques and methods to optimize generalization error and computational cost. In fact, recent developments in $W$-operator learning explore different techniques within the two-level design approach [35], [36], [37], [38].

### C. Window determination and selection

Recent window determination and selection methods for two level operators follow a same basic algorithm. Given a window domain $D$ and a set of training images, first a set of candidate windows, each contained in $D$, is created and then a subset of them to be used in the combination is selected.

In [38], a method called WER (Window selection using Entropy based Ranking) is proposed. First a set of candidate windows is manually defined. Then, the windows are ordered according to an improved conditional probability entropy, computed for each window over the set of all training window images collected from the training images. Next, operators are trained only for the $t$ top ranked windows, and then the two level operators are trained iteratively starting with the combination of the first two and adding to the group the next operator in the ordered sequence. The process stops when a combination with $t$ first level operators is processed and the combination with the best performance on a validation set is chosen.

The methods in [35], [36] employ feature selection techniques both to create windows and to select the windows to be used in the second level operator.

In [35], Santos *et al*. initially create first level windows using a grouping procedure that aims to decrease the Interaction Information [39] between pixels. Then they use a forward selection strategy based on a conditional mutual information to order the resulting operators. The number of operators to be combined is selected using a Minimum Description Length approach.

The FS method, described in [36], generates candidate windows by employing the Relief algorithm on the set of points in $D$. Relief ranks features (window points) based on a set of training instances (prototypes). Multiple window point rankings are created by randomly generating different sets of prototypes. Candidate windows are then created taking the 40 best features of each ranking and a $W$-operator is trained for each of them. These operators are also ranked by the same algorithm, setting them as the candidate features for the second level pattern. The $t$ best ranked operators are then used to train a second level operator, where $t$ is chosen in such a way as to control the amount of observed unique training second level patterns.

### D. Regularization based window determination

The methods presented in the section above have two weaknesses: (i) they do not jointly optimize the combination of windows and (ii) they are limited by the windows that are generated for the first level operators. The Near Infinitely Linear Combination (NILC) method [40] overcomes these weaknesses by simultaneously doing error minimization and feature selection. This is possible by learning a linear model using $L_1$ regularization.

Let $\mathcal{C} = \{W_i \subseteq D\}, |\mathcal{C}| = N = 2^{|D|}$ be the set of all subwindows of $D$ and $\psi_i$ be an operator trained with window $W_i$. We model $\Psi^{(2)}(z) : \{0,1\}^{2^D} \rightarrow \{0,1\}$ as a linear model $\Psi^{(2)}(z) = w^T z$, where $z \in \{0,1\}^N$ is a second level pattern ($z = (\psi_1(x|_{W_1}), \ldots, \psi_N(x|_{W_N}))$ ). We assume an ordering for all $N$ windows and set $z_0 = 1$, which means that the operator trained with the empty window always output white (1). The weights $w \in \mathbb{R}^N$ are learned by minimizing the cost function $c$ expressed as the cross-entropy loss [41] plus a $L_1$ regularization term ($\mathcal{Z}_{i\bullet} = z_i^T$):

$$\min_w c(w) = -\sum_{i=1}^{M}(y_i \mathcal{Z}_{i\bullet} w + log(1 + e^{\mathcal{Z}_{i\bullet} w}) + \lambda ||w||_1 \quad (12)$$

In this equation, $M$ is the number of training examples, $\mathcal{Z}_{i\bullet} = z_i^T$ is the $i^{th}$ second level pattern and $\lambda$ is the regularization parameter.

Directly optimizing Eq. 12 requires knowing in advance the full $\mathcal{Z}$ matrix. This is unfeasible, since it implies training an exponential number of operators. We address this challenge by adopting an active based method inspired by [42]. At the optimal solution $w^*$, the optimality conditions are:

$$-\sum_{i=1}^{M} \mathcal{Z}_{ij}(y_i - p_i) + \lambda = 0 \quad \text{if} \quad w_j^* > 0; \quad (13a)$$

$$-\sum_{i=1}^{M} \mathcal{Z}_{ij}(y_i - p_i) - \lambda = 0 \quad \text{if} \quad w_j^* < 0; \quad (13b)$$

$$-\lambda \leq -\sum_{i=1}^{M} \mathcal{Z}_{ij}(y_i - p_i) \leq \lambda \quad \text{if} \quad w_j^* = 0\,. \quad (13c)$$

where $p_i = \frac{e^{\mathcal{Z}_{i\bullet} w}}{1+e^{\mathcal{Z}_{i\bullet} w}}$.

Given a solution $w \in \mathbb{R}^N$ satisfying Eqs. 13a and 13b for all $w_i \neq 0$, we can check if adding a new operator $\psi_j$ to the combination decreases the cost by checking if Eq. 13c holds for $w_j$. In the positive case we can obtain a new solution by solving Eq.12 restricted to the indices $\{i|w_i \neq 0\} \cup \{j\}$. This induces an iterative algorithm (NILC) that, in each step, generates a new operator $\psi_j$, checks if it improves the current solution and updates it if necessary. The operators are generated by randomly sampling a window from $\mathcal{C}$ and training a $W$-operator in a separate training set.

Some advantages over previous methods are: (1) the regularization parameter $\lambda$ can be adjusted to include more or less operators in the combination, avoiding the need to specify the exact number of operators to be combined (Fig. 8 shows the impact, in the accuracy, of the number of operators in the combination); and (2) since at every step we have a valid solution to Eq. 12, we can continue an once stopped training if necessary. For instance, if slightly different new images are available, an already trained operator can be updated by just sampling new operators trained with the new images.

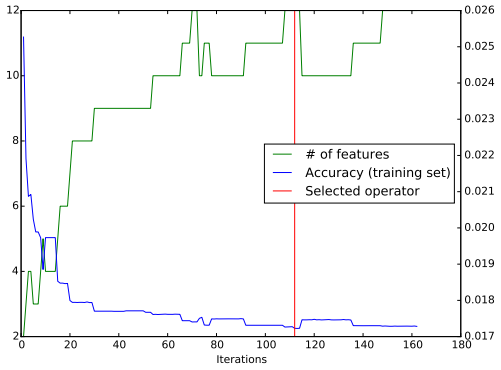



Figure 8: Evolution of the number of operators and error in the training set while NILC is running. The red line marks the operator selected by the validation procedure.

### *E. Learning over large windows using kernel approximations*

Kernel Support Vector Machines (kernel SVM) are known to generalize well even in high dimensional data and could be used to train $W$-operators on large windows. However, training kernelized SVMs may have prohibitively high computational and storage costs. In [43], we have used a recent technique called kernel approximation [44], [45], which computes an approximate solution to the kernel SVM problem. Given a kernel $k : \mathcal{X} \times \mathcal{X} \Rightarrow \mathbb{R}$, the main idea of kernel approximation is to find a feature map $\varphi : \mathcal{X} \rightarrow \mathbb{R}^N$ such that $k(x, x') \approx \varphi(x)^T \varphi(x')$. The Nyström method [44] computes this approximation from a (small) set of $m$ examples $\{x_i\}_{i=1}^m$ as

$$\varphi(x) = diag(\frac{1}{\sqrt{\lambda_j}}) {U^{(m)}}^T K_x, \quad (14)$$

where $K_x = (k(x, x_1), \ldots, k(x, x_m))$ and $\lambda_j$ and $U^{(m)}$ are the eigen values/vectors of the Gram matrix of all $x_i, i = 1, \ldots, m$.

Since binary and gray level inputs differ considerably in their distribution, we adopt different kernels in each case. For binary input images we use a polynomial kernel (Eq. 15).

$$K(x, x') = (x^T x' + c)^d\,. \quad (15)$$

where $d$ controls the degree of the polynomial and $c \leq 0$. We chose this kernel because the feature map for the polynomial kernel with degree $d$ contains all interaction terms of up to $d$ terms. Each interaction term can be interpreted as a pattern detector, since its result is 1 only when all variables are 1. This form also resembles the disjunctive normal form of a Boolean function presented in Section II.

In gray level images, many patterns that are visually very similar may actually differ in value for most of the pixels. Thus, a notion of similarity between patterns is necessary. Gaussian kernels (Eq. 16) can be interpreted as a soft threshold between two patterns, with the bandwidth $\gamma$ controlling the

tolerance of this measure[3].

$$K(x, x') = e^{-\gamma||x-x'||^2}. \quad (16)$$

A "bigger" $\gamma$ makes differences bigger, while a smaller one accepts more changes in the patterns. Also, since the Gaussian kernel is shift-invariant, it only takes into account the relative brightness between patterns. Therefore, this kernel is adequate to process gray level images.

The performance of the trained operators is affected by parameters $m$ (approximation size), $d$ (degree of the polynomial for the binary case) and $\gamma$ (bandwidth of the Gaussian kernel in the gray level case). These can be chosen using cross validation. An example of typical behavior for these parameters is shown in Figure 9. As the number of samples used to compute the approximation increases, the error tends to diminish. While for the polynomial kernel different values for degree $d$ do not result in significant differences in error, for the Gaussian kernel differences in the value of parameter $\gamma$ may result in large differences in error.

## V. TRIOSLIB – AN IMPLEMENTATION

`TRIOSlib` is a Python implementation of most of the methods presented in previous sections. It integrates with several scientific packages (such as Numpy and Scikit-learn) and it was built to allow new methods to be implemented without changing the library itself. TRIOSlib can be installed using `pip` by executing `$> pip install trios`.

As described in Section III-B, a $W$-operator consists of two parts: a feature extractor and a classifier. Instances of the `FeatureExtractor` class extract patterns from the images, which are then classified by an instance of `Classifier` class. The `WOperator` class combines these two components to train and transform images. Depending on the task being tackled, a different combination of feature extractor and classifier may be adequate.

The most basic feature extractor just copies the pixel values to a flat vector. More sophisticated techniques can be implemented coupling general feature extraction and pre-processing algorithms. For instance, window images for aperture operators are implemented as a feature extractor that subtracts the center pixel value from all other features and saturating them within the window range $[-r, r]$ when necessary.

In practice, any classifier method can be used to train a $W$-operator. TRIOSlib includes an implementation of the ISI algorithm [25] for Boolean function minimization. This algorithm was used in many earlier works [36], [37], [38], [35]. Other works [32], [40], [43] have used Decision Trees, which are available in TRIOSlib through a wrapper class (`SKClassifier`) for scikit-learn models.

A basic code sample using raw pixels as features and a decision tree as a classifier is shown below. Complete documentation is available at http://trioslib.sf.net.

```
from trios.classifiers import SKClassifier
from sklearn.tree import DecisionTreeClassifier
from trios.feature_extractors import \
        RAWFeatureExtractor
import trios
import numpy as np
import trios.shortcuts.persistence as p
# load training set
images = trios.Imageset.read('images/level1.set')
# create window
win = np.ones((5, 5), np.uint8)
# use Decision Tree Classifier
cls = SKClassifier(DecisionTreeClassifier())
# and raw pixels values as features
fext = RAWFeatureExtractor
op = trios.WOperator(win, cls, fext)
op.train(images)
# save trained operator
p.save_gzip(op, 'fname_here.op')
# and optionally load it later using p.load_gzip
img = p.load_image('images/jung-1a.png')
# Second argument is application mask.
out = op.apply(img, img)
p.save_image(out, 'teste.png')
```

Class `Imageset` is used to handle a collection of images. An operator (`op`) is an instance composed of a window, a feature extractor and a classifier. Method `train` is responsible for extracting the features from the images in the training set and then training the classifier. Method `apply` is responsible for applying a trained operator on an image. Application of the operator can be restricted to a subset of pixels in the image domain. In the code, the second parameter in call `op.apply(img, img)` means that the operator will be applied only on the foreground pixels. This is useful when one knows that the desired operator is anti-extensive (i.e., $\Psi(S) \subseteq S$).

## VI. APPLICATION EXAMPLES

Due to length restrictions, the examples are presented in a summarized form. A more complete description regrading datasets, parameter values, scripts and results can be found in the cited references and in the TRIOSlib [46] companion website.

### A. Document processing

Recognizing specific elements, such as text, figures, logo, handwriting, and others, in document images is an important step in document analysis. We illustrate the application of trained $W$-operators on two document processing tasks: text segmentation (*TexRev* dataset) and character recognition (*CharS* dataset). These datasets have been used as a benchmark for $W$-operator learning in previous works [30], [35], [38], [40], [43].

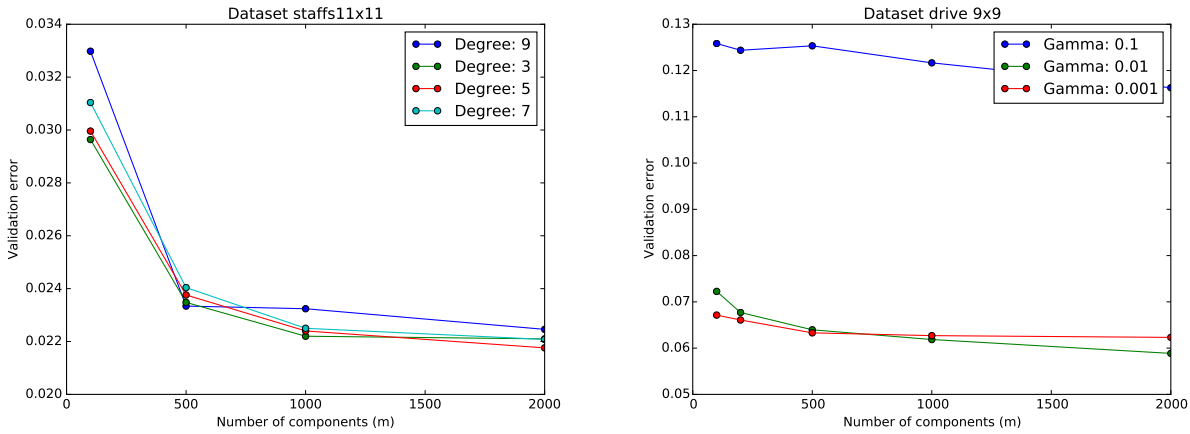
In Table I we show the MAE of two-level operators obtained with different methods, computed on a common test set. First row ("Manual") refers to manual definition of both the number of operators to be combined and the individual windows. This example illustrates the evolution obtained along the years.

A typical result for character recognition is shown in Fig. 10 and for text segmentatin in Fig. 11.

### B. Staff Removal

Staff removal is a common task in Optical Music Recognition systems. It consists in removing the stafflines from music

[3]If the patterns are similar, the norm is small, so the kernel result.




(a) (b)

Figure 9: Validation error w.r.t. to changes in the kernel's parameters. (a) Polynomial kernel; (b) Gaussian kernel.

| Domain | Method | *CharS* | *TexRev* |
|---|---|---|---|
| $9 \times 7$ | Manual [30] (2009) | 0.0050 | 0.026 |
| | WER [38] (2015) | 0.0046 | 0.022 |
| | NILC [40] (2016) | 0.0045 | **0.017** |
| | KA [43] (2016) | **0.0042** | 0.024 |

Table I: Comparative test errors for datasets *CharS* and *TexRev*.

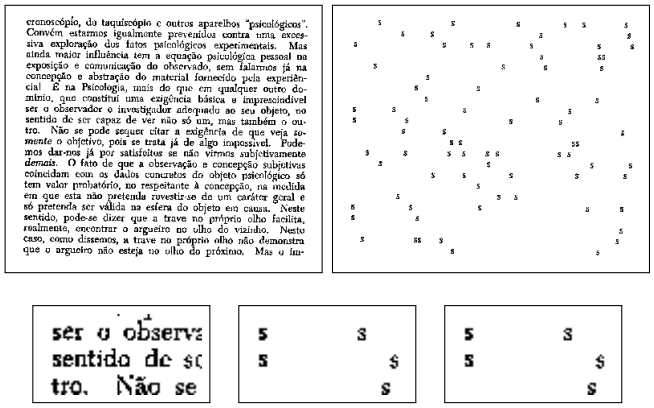


Figure 10: Typical results for images in the *CharS* dataset. At the top, a pair of input and expected output images. At the bottom, from left to right, highlight of a region of the input test, expected output, and resulting images.

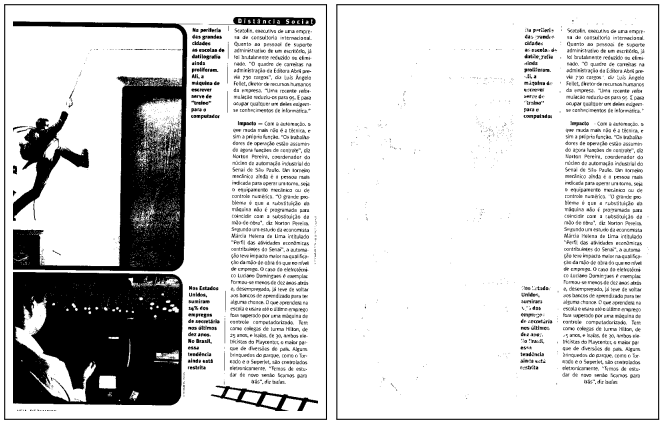


Figure 11: A typical result for images in *TexRev* dataset. Input test image (left) and resulting image (right).

| Method | Acc (%) | Rec (%) | Spec (%) | F1 |
|---|---|---|---|---|
| LTC [47] | 87.58 | 64.66 | **99.53** | 80.40 |
| Skeleton [47] | 94.50 | 86.97 | 99.03 | 92.24 |
| LRDE [48] | 97.03 | 94.02 | 98.84 | 95.97 |
| FS [37] | **97.34** | **95.87** | 98.22 | 95.90 |
| Manual [36] | 96.89 | 94.37 | 98.41 | 95.51 |
| NILC [40] | 97.10 | 94.58 | 98.61 | **96.07** |
| KA [43] | 96.23 | 92.85 | 98.26 | 94.88 |

Table II: Comparative results, accuracy (Acc), recall (Rec), specificity (Spec) and F1-measure (F1), for the *staffs* dataset.

scores (part of a pair of training images is shown in Fig. 1, at the beginning of this text). Existing solutions usually employ heuristic methods. Machine learning based methods are more attractive to cope with image variations [36], [37], [40].

Table II compares the performance of trained image operators (using FS, NILC and KA methods and also with a manually designed set with the seven reference windows shown in Fig. 12) with some of the state-of-the-art heuristic methods (LTC [47], Skeleton [47] and LRDE [48]) on a test set of the CVC-MUCIMA-2013 dataset [49]. $W$-operators outperforms most heuristic methods. Even the operator based on the set of reference windows obtained good results.

Samples of the resulting images are shown in Fig. 13.

### C. Vessel Segmentation

This example illustrates the application of image operator learning for gray-scale image processing. The DRIVE dataset [50] contains input and groundtruth images for the segmentation of blood vessels in retina images. Method KA

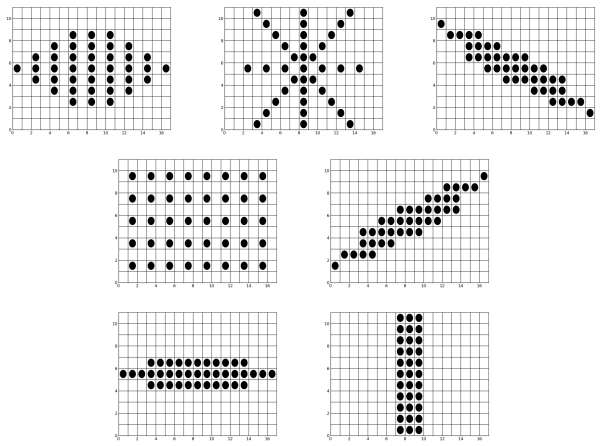


Figure 12: Set of reference windows for staff removal [36].

| Method | Accuracy (%) |
|---|---|
| Staal [50] | **9**4.42 |
| Niemeijer [51] | 94.16 |
| Zana [52] | 93.77 |
| KA | 93.91 |

Table III: Results for the *DRIVE* dataset.

is compared with three heuristic methods in Table III. The parameters for KA were determined using a validation set.

Some result images are shown in Figure 14. All tested methods missed the very thin vessels. Staal had better overall performance being consistent with the results in Table III. Nevertheless, note that the output of KA is mostly connected even though each pixel is classified independently from the others. This is a encouraging result that indicates that additional adjustments (for instance, other types of kernel) may lead to better results.

## VII. Conclusion

Image operator learning aims to mitigate image processing tasks that demand knowledge and experience on building pipelines of image operators and operations. The learning framework is based on a theoretically solid foundation, namely the characterization of image operators by local functions, and a statistical estimation model that enables the formulation of optimal image operator estimation as a classifier training problem.

Learning methods may exploit and combine representation structures such as the two level composition of operators and machine learning techniques such as feature selection and kernel learning. This tutorial summarized almost 25 years of research in the area and presented an updated open source implementation of the software to make the work easily reproducible.

## Acknowledgments

This work is supported by FAPESP (2015/17741-9, 2011/50761-2,2015/01587-0) and CNPq (484572/2013-0). Igor S. Montagner is supported by FAPESP (2014/21692-0, 2011/23310-0). N. S. T. Hirata and R. Hirata Jr. are partially supported by CNPq.

## References

[1] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012, pp. 1106–1114.

[3] Google. (2016, April) Google self-driving car project. [Online]. Available: https://www.google.com/selfdrivingcar/

[4] S. D. J. Prince, *Computer Vision – Models, Learning and Inference*. Cambridge, 2012.

[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2010, pp. 253–256.

[8] R. Szeliski, *Computer Vision – Algorithms and Applications*. Springer, 2011.

[9] H. J. A. M. Heijmans, *Morphological Image Operators*. Boston: Academic Press, 1994.

[10] P. Soille, *Morphological Image Analysis*, 2nd ed. Berlin: Springer-Verlag, 2003.

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

[12] E. R. Dougherty, "Optimal mean-square N-observation digital morphological filters: I. Optimal binary filters," *CVGIP: Image Understanding*, vol. 55, no. 1, pp. 36 – 54, 1992.

[13] J. Yoo, K. L. Fong, J.-J. Huang, E. J. Coyle, and G. B. Adams III, "A Fast Algorithm for Designing Stack Filters," *IEEE Transactions on Signal Processing*, vol. 8, no. 8, pp. 1014–1028, August 1999.

[14] E. J. Coyle and J.-H. Lin, "Stack Filters and the Mean Absolute Error Criterion," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no. 8, pp. 1244–1254, August 1988.

[15] I. Tosic and P. Frossard, "Dictionary learning," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 27–38, March 2011.

[16] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press, 2004.

[17] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, 2004.

[18] J. Serra, *Image Analysis and Mathematical Morphology. Volume 2: Theoretical Advances*. Academic Press, 1988.

[19] G. J. F. Banon and J. Barrera, "Decomposition of Mappings between Complete Lattices by Mathematical Morphology, Part I. General Lattices," *Signal Processing*, vol. 30, pp. 299–327, 1993.

[20] ——, "Minimal Representations for Translation-Invariant Set Mappings by Mathematical Morphology," *SIAM J. Applied Mathematics*, vol. 51, no. 6, pp. 1782–1798, December 1991.

[21] G. Matheron, *Random Sets and Integral Geometry*. John Wiley, 1975.

[22] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image Analysis Using Mathematical Morphology," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 4, pp. 532–550, July 1987.

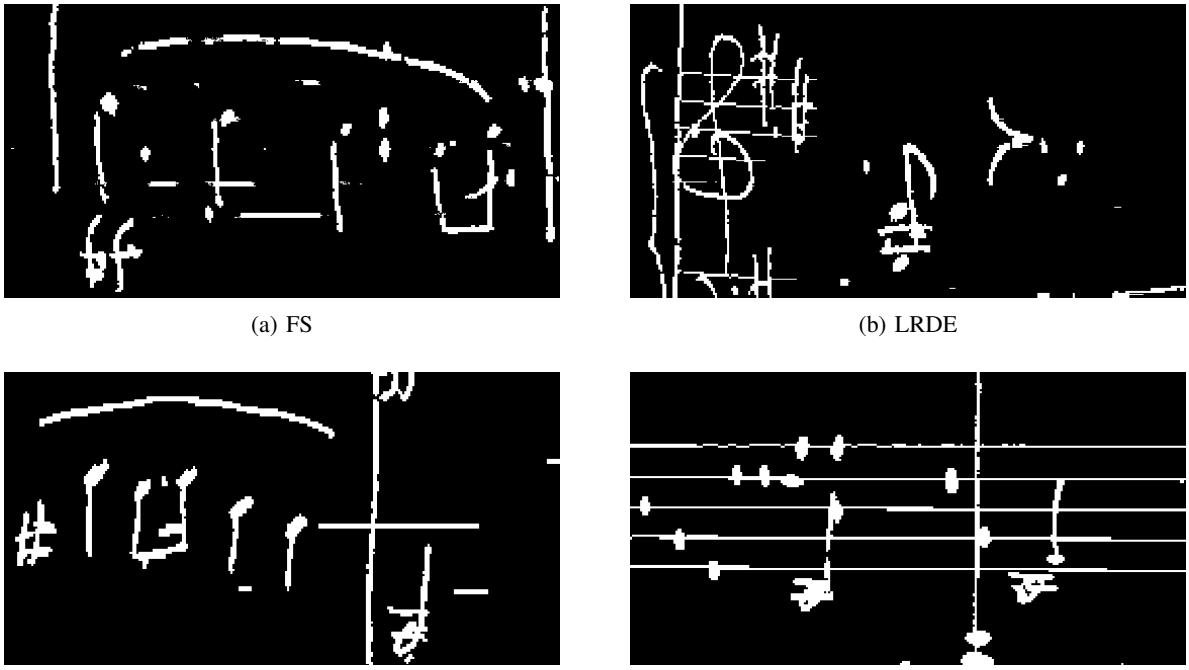
[23] E. R. Dougherty and R. A. Lotufo, *Hands-on Morphological Image Processing*. SPIE Press, 2003.

[24] J. Barrera, R. Terada, R. J. Hirata, and N. Hirata, "Automatic programming of morphological machines by pac learning," *Annales Societatis Mathematicae Polonae. Series 4: Fundamenta Informaticae*, vol. 41, pp. 229–258, 2000.

[25] N. S. T. Hirata, R. Hirata Jr., and J. Barrera, "Basis computation algorithms," in *Mathematical Morphhology and its Applications to Signal and Image Processing (Proceedings of the 8th International Symposium on Mathematical Morphology)*, 2007, pp. 15–26.

[26] B. Naegel, N. Passat, and C. Ronse, "Grey-level hit-or-miss transforms – Part I: Unified theory," *Pattern Recogn.*, vol. 40, no. 2, pp. 635–647, 2007.

[27] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.



(a) FS (b) LRDE

(c) Skeleton (d) LTC

Figure 13: Frequent errors made by different staff removal methods.


[28] P. D. Wendt, E. J. Coyle, and N. C. Gallagher Jr., "Stack Filters," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-34, no. 4, pp. 898–911, August 1986.

[29] J. Barrera, E. R. Dougherty, and N. S. Tomita, "Automatic Programming of Binary Morphological Machines by Design of Statistically Optimal Operators in the Context of Computational Learning Theory," *Electronic Imaging*, vol. 6, no. 1, pp. 54–67, January 1997.

[30] N. S. T. Hirata, "Multilevel training of binary morphological operators," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 707–720, April 2009.

[31] R. Hirata Jr., E. R. Dougherty, and J. Barrera, "Aperture Filters," *Signal Processing*, vol. 80, no. 4, pp. 697–721, April 2000.

[32] R. Hirata Jr., M. Brun, J. Barrera, and E. R. Dougherty, "Aperture filters: theory, application, and multiresolution analysis," in *Advances in Nonlinear Signal and Image Processing*, ser. EURASIP Book Series on Signal Processing and Communications, S. Marshall and G. L. Sicuranza, Eds. Hindawi, 2006, vol. 6, pp. 15–48.

[33] D. Dellamonica Jr., P. J. S. Silva, C. Humes Jr., N. S. T. Hirata, and J. Barrera, "An Exact Algorithm for Optimal MAE Stack Filter Design," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 453–462, 2007.

[34] J. Barrera and E. R. Dougherty, "Representation of gray-scale windowed operators," in *Mathematical Morphology and its Applications to Image and Signal Processing*, H. Heijmans, Ed. Kluwer Academic Publishers, June 1998, pp. 19–26.

[35] C. S. Santos, N. S. Hirata, and R. Hirata Jr., "An information theory framework for two-stage binary image operator design," *Pattern Recognition Letters*, vol. 31, no. 4, pp. 297–306, 2010.

[36] I. Montagner, R. Hirata Jr., and N. S. T. Hirata, "A machine learning based method for staff removal," in *22nd International Conference on Pattern Recognition (ICPR)*, 2014, pp. 3162 – 3167.

[37] ——, "Learning to remove staff lines from music score images," in *IEEE International Conference on Image Processing 2014 (ICIP 2014)*, Paris, France, Oct. 2014, pp. 2614 – 2618.

[38] M. Dornelles and N. S. T. Hirata, "Selection of Windows for W-Operator Combination from Entropy Based Ranking," in *Conference on Graphics, Patterns and Images (SIBGRAPI)*, Aug 2015, pp. 64–71.

[39] A. Jakulin and I. Bratko, "Quantifying and visualizing attribute interactions," *CoRR*, vol. cs.AI/0308002, 2003. [Online]. Available: http://arxiv.org/abs/cs.AI/0308002

[40] I. S. Montagner, N. S. T. Hirata, R. Hirata, and S. Canu, "NILC: a two level learning algorithm with operator selection," in *IEEE International Conference on Image Processing (ICIP)*, 2106, pp. 1873–1877.

[41] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer, 2009.

[42] R. Flamary, F. Yger, and A. Rakotomamonjy, "Selecting from an infinite set of features in SVM," in *European Symposium on Artificial Neural Networks*, 2011.

[43] I. S. Montagner, S. Canu, N. S. T. Hirata, and R. Hirata Jr., "Kernel Approximations for $W$-operator learning," in *To appear in Proceedings of SIBGRAPI 2016 (XXIX Conference on Patterns, Graphics and Images)*, 2016.

[44] C. K. I. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. MIT Press, 2001, pp. 682–688.

[45] A. Rahimi and B. Recht, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2009, pp. 1313–1320.

[46] "Trios - training image operators from samples," Jun. 2012. [Online]. Available: https://sourceforge.net/projects/trioslib/

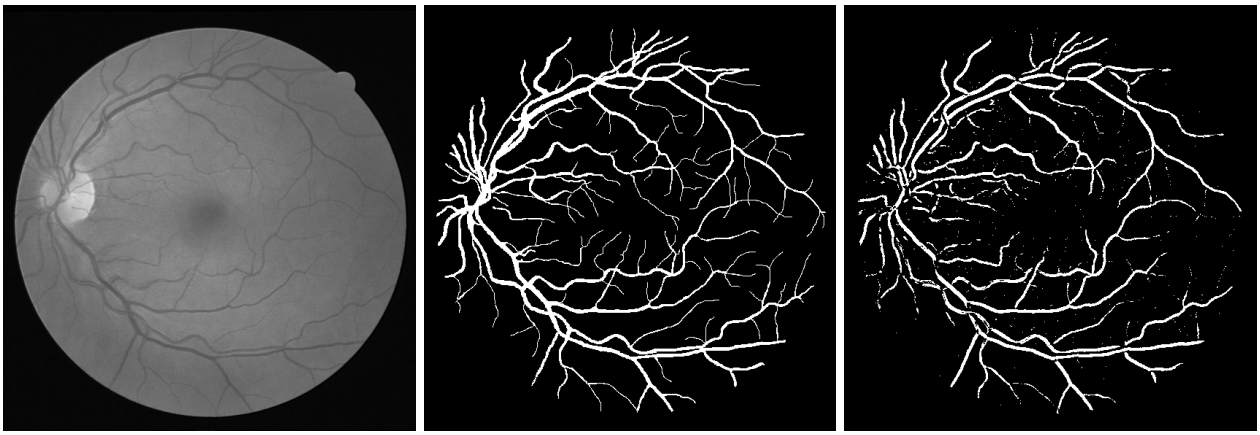
[47] C. Dalitz, M. Droettboom, B. Pranzas, and I. Fujinaga, "A comparative study of staff removal algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 753–766, 2008.

[48] T. Géraud, "A morphological method for music score staff removal," in *Proceedings of the 21st International Conference on Image Processing (ICIP)*, Paris, France, 2014, pp. 2599–2603.

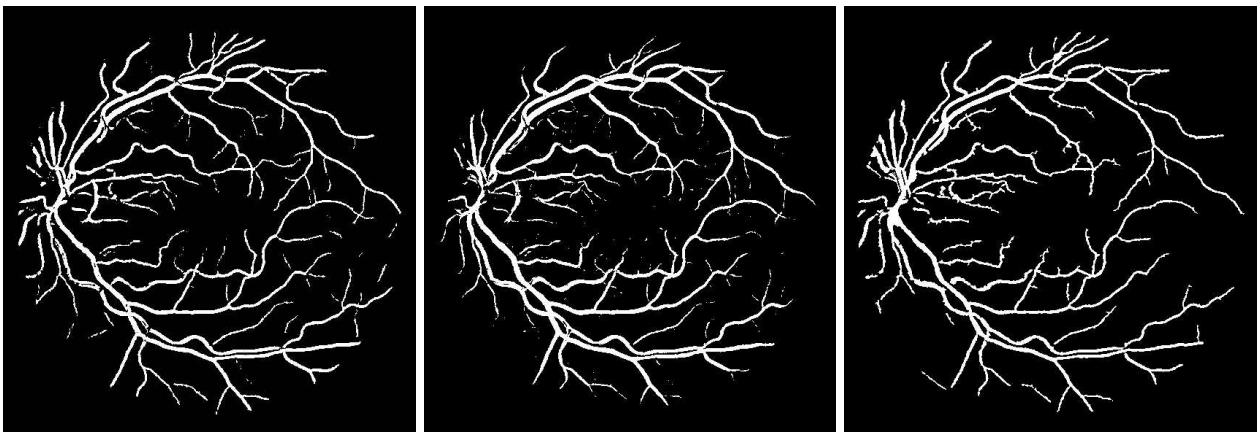
[49] M. Visaniy, V. Kieu, A. Fornes, and N. Journet, "ICDAR 2013 music scores competition: Staff removal," in *12th International Conference on Document Analysis and Recognition (ICDAR),*, 2013, pp. 1407–1411.

[50] J. Staal, M. Abramoff, M. Niemeijer, M. Viergever, and B. van Ginneken, "Ridge based vessel segmentation in color images of the retina," *IEEE Transactions on Medical Imaging*, vol. 23, no. 4, pp. 501–509, 2004.

[51] M. Niemeijer, J. Staal, B. van Ginneken, M. Loog, and M. Abramoff, "Comparative study of retinal vessel segmentation methods on a new



(a) Input (b) Expected output (c) Our method (KA)



(d) Staal (e) Niemeijer (f) Zana

Figure 14: Result images for the *DRIVE* dataset.

publicly available database," in *SPIE Medical Imaging*, J. M. Fitzpatrick and M. Sonka, Eds., vol. 5370. SPIE, 2004, pp. 648–656.

[52] F. Zana and J.-C. Klein, "Segmentation of vessel-like patterns using mathematical morphology and curvature evaluation," *IEEE Transactions on Image Processing*, vol. 10, no. 7, pp. 1010–1019, 2001.