IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 28, 2018, accepted July 9, 2018, date of publication July 23, 2018, date of current version August 15, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2858660

Evolutionary Design of Problem-Adapted Image Descriptors for Texture Classification

VALENTÍN CALZADA-LEDESMA[1], HÉCTOR J. PUGA-SOBERANES[1], MANUEL ORNELAS-RODRÍGUEZ[1], ALFONSO ROJAS-DOMÍNGUEZ[1], JUAN MARTÍN CARPIO-VALADEZ[1], ANDRÉS ESPINAL[2], JORGE A. SORIA-ALCARAZ[2], AND MARCO A. SOTELO-FIGUEROA[2]

[1]Tecnológico Nacional de México, Instituto Tecnológico de León, 37290 León, Mexico
[2]Organizational Studies Department, University of Guanajuato, 36000 Guanajuato, Mexico

Corresponding author: Alfonso Rojas-Domínguez (alfonso.rojas@gmail.com)

The work of V. Calzada-Ledesma was supported by the National Council of Science and Technology of Mexico under Grant 263129. The work of A. Rojas-Domínguez was supported by the National Council of Science and Technology under Grant CATEDRAS-2598.

ABSTRACT Effective texture classification requires image descriptors capable of efficiently detecting, extracting, and describing the most relevant information in the images, so that, for instance, different texture classes can be distinguished despite image distortions such as varying illuminations, viewpoints, scales, and rotations. Designing such an image descriptor is a challenging task that typically involves the intervention of human experts. In this paper, a general method to automatically design effective image descriptors is proposed. Our method is based on grammatical evolution and, using a set of example images from a texture classification problem and a classification algorithm as inputs, generates problem-adapted image descriptors that achieve very competitive classification results. Our method is tested on five well-known texture data sets with different number of classes and image distortions to prove its effectiveness and robustness. Our classification results are statistically compared against those obtained by means of six popular hand-crafted texture descriptors in the state of the art. This statistical analysis shows that our evolutionarily designed descriptors outperform most of those designed by human experts.

INDEX TERMS Image texture analysis, image classification, classification algorithms, genetic programming, evolutionary computation.

I. INTRODUCTION

Computer Vision (CV) applications frequently deal with visual features known as textures; these may be present in images as natural or human-made objects, such as clouds, sand, mosaic, carpet, etc. Basically, a texture is a property of a region which can be easily recognized by a human through visual inspection. However, from the point of view of digital image processing there is not a generally accepted formal definition of texture [42]. According to some authors [55], [61], there are two main approaches to defining texture. Under the Structural approach, texture is a set of primitives, often called texels or textons, arranged in some regular or repeated spatial relationship. According to the Statistical approach, texture is a quantitative measure computed from the gray-level intensities of an image. While the first approach may describe well some human-made textures, the second approach is more general, computationally efficient, and is used more often in practice. Dealing with textures is a challenging task; several techniques have been proposed to deal with texture-related problems, such as searching for anomalies in a specific texture, splitting an image into different regions, and recognizing what a region represents based on examples of labeled textures [51]. In this work, we discuss this latter problem, known as texture classification.

Solving texture classification problems involves solving classification tasks mainly defined around three spaces [8]:

- The Pattern Space, with dimensionality R, consists of digital images captured from the real world.
- The Feature Space, an intermediate space, consists of simpler abstractions of the elements in the Pattern Space, and usually it has a dimensionality $N \ll R$; such dimensionality reduction may result from selecting the most representative characteristics of images to construct a feature vector.
- The Classification Space, with dimensionality K (number of classes in the problem), is generated by means

of a classification algorithm that produces a partition of the feature space.

Although classifiers partitioning the feature space can be based on different approaches (such as mathematical, statistical, neural-inspired, etc.), the *No-Free-Lunch Theorem* states that no single classification algorithm can achieve the top performance on every problem [29], [62]. This implies that the definition of the feature space may play an important role, affecting a classifier's performance for a specific problem. Unfortunately, it is not always clear which features should be used to form the feature space. It is desirable that features simultaneously be discriminative and robust (offering invariance to image distortions such as changes in illumination, viewpoint, rotation, etc.), but this is not easy to achieve. Thus, the motivation behind the present study is to develop a methodology for automatic feature extraction.

Texture descriptors have been specifically designed to perform the feature extraction process, detecting the visual characteristics of textures and allowing their classification [5]. The design of said descriptors is a subject of ongoing research, given that there is no single descriptor capable of representing all types of textures effectively and robustly.

Currently, there is a growing trend towards the development of preprocessing and transforming data methods to enhance pattern representation and to improve pattern classification [15], [57]. Several of these methods avoid the intervention of human experts to some degree; these can use neural network-based approaches such as in the Deep Learning (DL) [31] paradigm, or optimization methods such as Genetic Programming (GP) [38] and Grammatical Evolution (GE) [52], among others. The DL approaches have been successfully applied to extract machine-learned features for image classification problems (e.g. object recognition) [15], but recent studies show that texture classification with handcrafted descriptors, such as Local Binary Patterns (LBP) and Haralick's texture descriptors can equal or surpass the performance of DL approaches [12], [13], [19], [39]. Simpler descriptors signify lower computational costs than highly-dense-connected neural networks; besides, DL approaches also require a considerably large number of examples for their training.

Regarding optimization methods, stochastic search and machine learning algorithms are typically used in combination in order to improve classification performance by means of identifying optimum feature subsets, from a given set of features proposed by human experts. In this case, the specific classification algorithm is involved as part of the optimization function used to drive the optimization process [10], [48]. A recent proposal consists in combining several primitive operators through GP and Multi-Objective GP (MOGP) to artificially synthesize image descriptors capable of dealing with tasks such as object recognition [49], [50], object detection [23], and interest point detection [43], [44], [60]. MOGP algorithms have also been used to generate holistic descriptors for scene recognition problems [40], [54]. In contrast to the numerous efforts devoted to generating image descriptors for object and scene recognition, there are considerably fewer works proposing the use of GP algorithms for texture classification [2]–[5], [37].

The main contribution of this work is a GE-based methodology that automatically generates texture descriptors optimized for a given classification problem while only requiring as inputs the texture classification problem and a specific classification algorithm. During the evolutionary process, candidate texture descriptors that detect and extract features of the texture images are generated and evaluated. The extracted features are contained in a histogram-based feature vector, which is a simple, fast to calculate, and commonly used texture representation [21]. The output of our proposed methodology is a problem-adapted texture descriptor generated without prior knowledge of the dataset and without involving a human expert, but evolved to favor classification performance on the given problem.

The rest of this paper is organized as follows. Section II describes the proposed methodology and related concepts. Section III contains the experimental design, the datasets description and parameter settings. Experimental results are reported and discussed in Section IV. Finally, our conclusion and directions for future work are offered in Section V.

II. PROPOSED METHODOLOGY AND CONCEPTS

In general, an adequate feature representation, or feature set, is essential to achieving high performance in a classification task. However, it is often not easy to determine the most suitable feature set for a given problem. A common practice is to use state-of-the-art texture descriptors that have been tested in similar kinds of problems, but there is no guarantee that such descriptors will work well on the new texture problems. Besides, it may be necessary to identify the appropriate classifier to deal with the proposed feature space.

The purpose of our proposed methodology is to automatically generate problem-adapted texture descriptors guided by the classification performance on said problem. The evolved descriptors detect and exalt texture features represented as a histogram-based feature vector, thus achieving a significant dimensionality reduction, from the total number of pixels in an image to a feature space of only 2^b dimensions, where b is the image's color-depth.

The advantage of our methodology is that it enables the automatic design of texture descriptors without requiring prior knowledge of the particular texture problem, thus dispensing with a human expert. Additionally, our methodology is implemented under a modular design, producing a feature extractor that is not attached to any particular classifier (similar strategies are followed in [2]–[4], [43], [44], [49], and [50]). Fig. 1 shows a diagram of our proposed methodology, where one can observe that the descriptor is generated as a program, defined by an ad-hoc language. This methodology is based on Grammatical Evolution (GE, Section II-A). The required inputs are: a sample set of texture images from the classification problem and a classifier (gray elements in Fig. 1).

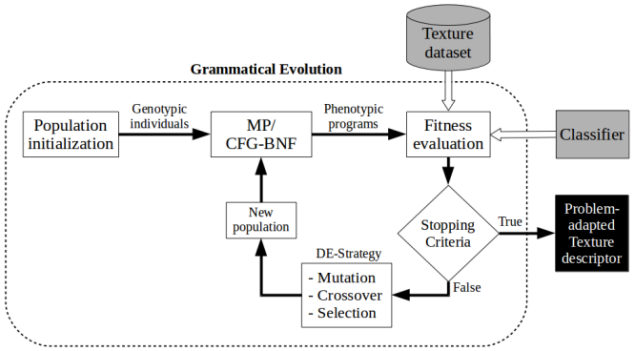


FIGURE 1. Methodology for automatic design of texture descriptors.

By means of GE (represented by the dotted rectangle), our proposal generates a problem-adapted texture descriptor (black element in Fig. 1). The following sections describe in detail each of the elements required by our proposal.

A. GRAMMATICAL EVOLUTION

Grammatical Evolution (GE) [52], a grammar-based form of Genetic Programming (GP) [38], automatically creates syntactically correct programs, according to a language related to the kind of problem at hand. The GE is built around three main elements [58]: a *Context-Free Grammar in Backus-Naur Form* (CFG-BNF, Section II-B) as a way to represent programs; a *Search Engine* (Section II-C) to explore the program space looking for the optimal program; and a *Mapping Process* (MP, Section II-D) to 'derive' or decode programs. The quality of each program is evaluated by means of a fitness function (Section II-E).

Several variants of GE have been proposed, according to the search engine that is employed. In this work, the Differential Evolution Algorithm is used as the search engine; this configuration is known as Grammatical Differential Evolution [45] (GDE). GDE achieves a more compact representation of solutions, compared to the original GE technique that employs a Genetic Algorithm as search engine. This is because GDE can handle real values instead of only binary ones. GDE also has fewer parameters than the original GE and Grammatical Swarm [47] (which uses Particle Swarm Optimization as search engine).

The GDE workflow in this work is as follows. First, a population of candidate descriptors is created and initialized, with each candidate descriptor being represented by an n-dimensional vector of real values called codons; this vector is a descriptor's genotypic-form, which is transformed into its phenotypic-form (or functional form) by means of the CFG-BNF and an MP. Once the phenotypic-form of a candidate descriptor has been obtained, its quality (performance on the given problem) is measured through the fitness function. Then the DE module applies different evolutionary operators (mutation, crossover and selection) on the population of descriptors with the objective of eventually producing a descriptor that solves the texture problem optimally. The process produces a new population and the cycle is repeated until one or more stopping criteria are satisfied.

B. CONTEXT-FREE GRAMMAR IN B-N FORM

A Context-Free Grammar in Backus-Naur Form is commonly employed to define a language used to validate the syntax of a program, but is also used in GE to design programs defined by the rules in it [25]. A CFG-BNF is made up of the 4-tuple $\{N, T, P, S\}$; where N is the set of *Non-terminal symbols* that can be expanded into one or more Terminals and Non-terminals; T is the set of *Terminal symbols*, i.e. the accepted alphabet employed to generate valid programs; P represents the set of *Production rules* that maps $N \rightarrow T$, and $S \in N$ represents an initial symbol.

In this work, the proposed CFG-BNF provides the structure of the language required to define texture descriptors. The descriptors are generated using infix notation and these are formed by the combination of primitive operators: High-pass filters, Low-pass filters, Directional filters, Max-pooling filters and Arithmetic operators. These primitive operators are described below.

High-Pass Filters: can highlight areas of greater variability in an image. In our proposal, the Laplacian, Laplacian of Gaussian, and Gaussian derivatives are used.

Low-Pass Filters: their purpose is to reduce the noise in images by attenuating the high-frequency information. In this work, the Gaussian, Average and Median filters are used.

Directional Filters: a Gabor Filter Bank (GFB) is a technique widely-used to analyze an image in its spatial and frequency domains by applying a set of coordinated Gabor filters [16]. In this work, GFBs composed of six Gabor filters each, are employed. The filters in a bank correspond to different scales but the same orientation, as in [40] and [54]. Thus, a GFB applied to an image results in six convolved images, one per filter; then the highest intensity value is selected across them element-wise, producing a final image. The configurations of the GFBs are shown in Table 1.

TABLE 1. Specific configurations of the Gabor filter bank.

Quantity	Value
Wavelength	7.0
Aspect Ratio	0.5
Phase Offset	0
Bandwidth	1
Scales	$k \times k$, $k \in \{5, 7, 9, 11, 13\}$ pixels
Orientations per GFB	$0°, 45°, 90°, 135°$

Max-Pooling Filters: these filters return the maximum value inside a given region of size $n \times n$ pixels. Their purpose is to reduce the size of the representation (image size) while preserving the most relevant information [17], [53]. In this work, the Max-pooling filter is applied with different window sizes and a stride $= 2$. If needed (e.g. to allow all operations to be performed on images of the same size), the images are re-scaled by bilinear interpolation [54].

In addition to the use of image filters, basic arithmetic operations between primitive operators are employed during the evolutionary process that allow the generation of texture descriptors capable of extracting multiple low-level features from the images. Also, histogram equalization is used for contrast adjustment, and intensity scaling (multiply an image by 0.5 element-wise) is used for brightness adjustment.

The design of the CFN-BNF, made up by the 4-tuple $\{N, T, P, S\}$, is discussed below. Notice that throughout these sets of symbols and production rules, the closure property is satisfied: every operator and function returns an image that can become the input to any other operator or function.

The set of Non-terminal symbols (N) can be seen as a categorization of the primitive operators:

$$N = \{\langle Start\rangle, \langle MP\rangle, \langle Expr\rangle, \langle Filter\rangle, \langle Gau\rangle, \langle Lap\rangle, \langle GFP\rangle, \langle Arith\rangle, \langle op\rangle, \langle Terminal\rangle\}$$

The set of Terminal symbols (T) includes all the symbols accepted for the generation of a valid texture descriptor:

$$T = \{MP2, MP4, MP6, MP8, MP10, Gau1, Gau2, GauDX, GauDY, Lap, LapG1, LapG2, GFB0, GFB45, GFB90, GFB135, AverF, MedF, HEq, AbsV, Sqr, Sqrt, Log, T0.5, Ig, +, -, a-, *, /, (,)\}$$

A description of each terminal symbol and the parameters setting for each primitive operator is shown in Table 2.

The production rules (P) are defined as follows:

$$\begin{aligned}
P = \{&\langle Start\rangle ::= \langle MP\rangle\,(\langle Expr\rangle)\\
&\langle MP\rangle ::= MP2\,|MP4|\,MP6\,|MP8|\,MP10\\
&\langle Expr\rangle ::= (\langle Expr\rangle\langle Op\rangle\langle Expr\rangle)\,|\\
&\qquad \langle Filter\rangle\,(\langle Expr\rangle)\,|\langle Terminal\rangle\\
&\langle Filter\rangle ::= \langle Gau\rangle\,|\langle Lap\rangle|\,\langle GFB\rangle|\langle Arith\rangle\\
&\langle Gau\rangle ::= Gau1|Gau2\,|GauDX|\,GauDY\\
&\langle Lap\rangle ::= LapG1\,|LapG2|\,Lap\\
&\langle GFB\rangle ::= GFB0\,|GFB45|\,GFB90|GFB135\\
&\langle Arith\rangle ::= AverF\,|MedianF|\,HEq|\\
&\qquad AbsV\,|Sqr|\,Sqrt\,|Log|\,T0.5\\
&\langle Op\rangle ::= +\,|-|\,a-\,|*|\,/\\
&\langle Terminal\rangle ::= Ig\}
\end{aligned}$$

where there are N_r production rules that can be applied for each Non-terminal. These rules are separated by a "|" (OR) symbol and are enumerated from 0 to $N_r - 1$. Finally, the initial symbol, S, is defined as $\langle Start\rangle$.

C. DIFFERENTIAL EVOLUTION SEARCH ENGINE

Differential Evolution is used in this work as a Search Engine because it can be easily applied to a wide variety of problems with continuous variables and often shows better results than a Genetic Algorithm and other evolutionary algorithms. In addition, the DE parameters do not require the same fine tuning which is necessary for many other evolutionary algorithms [34], [63]. Although there are several variants of DE, this proposal is based on the *DE/rand/1/bin scheme* [59]. A detailed explanation follows.

TABLE 2. Terminal symbols and parameters of the primitive operators.

Symbol	Description (sizes are in pixels)
$MP2$	Max-pooling with a window of size 2 × 2 [54]
$MP4$	Max-pooling with a window of size 4 × 4 [54]
$MP6$	Max-pooling with a window of size 6 × 6 [54]
$MP8$	Max-pooling with a window of size 8 × 8 [54]
$MP10$	Max-pooling with a window of size 10 × 10 [54]
$Gau1$	Gaussian filter with $\sigma = 1$ [44]
$Gau2$	Gaussian filter with $\sigma = 2$ [44]
$GauDX$	The derivative along the horizontal axis, X
$GauDY$	The derivative along the vertical axis, Y
Lap	Laplacian filter
$LapG1$	Laplacian of Gaussian filter with $\sigma = 1$ [44]
$LapG2$	Laplacian of Gaussian filter with $\sigma = 2$ [44]
$GFB0$	Gabor Filter Bank with orientation of 0º [54]
$GFB45$	Gabor Filter Bank with orientation of 45º [54]
$GFB90$	Gabor Filter Bank with orientation of 90º [54]
$GFB135$	Gabor Filter Bank with orientation of 135º [54]
$AverF$	Average Filter with a window of size 5 × 5 [54]
$MedF$	Median Filter with a window of size 5 × 5 [54]
HEq	Histogram Equalization
$AbsV$	The Absolute Value of an image[a]
Sqr	The Square of an image[a]
$Sqrt$	The Square Root of an image[a]
Log	The base-2 Logarithm of an image[a]
$T0.5$	Multiply image by 0.5[a]
Ig	Obtain the gray-level image
$+$	Add two images[a]
$-$	Subtract two images[a]
$a-$	Absolute subtraction of two images[a]
$*$	Multiply two images[a]
$/$	Divide two images, using protected division[a]
(	Left parenthesis used for operation precedence
)	Right parenthesis used for operation precedence

[a]These are element-wise operations.

At the start of the DE algorithm, a population $\mathbb{P}$ formed by N_P individuals of size L is uniformly and randomly generated across the search space and evaluated using a fitness function. For each individual (referred to as a target vector) in $\mathbb{P}$, a 'mutant vector' ω is generated using the *Mutation* operation defined in (1), where ε_1, ε_2, and ε_3 are three different individuals randomly selected in the current generation (g), and $F \in (0, 2]$ is a constant factor that controls the amplification of the differential variation.

$$\omega = \varepsilon_1 + F \cdot (\varepsilon_2 - \varepsilon_3) \tag{1}$$

Next, a trial vector γ is formed using the *Crossover* operation defined in (2), where ind_j indicates the j-th component of the target vector, with $j \in \{1, \ldots, L\}$; $r\ and_j \in [0, 1)$ is a uniform random number and $C_r \in [0, 1]$ is a crossover rate.

$$\gamma_j = \begin{cases} \omega_j & \text{if } rand_j \leq C_r \\ ind_j & \text{otherwise} \end{cases} \tag{2}$$

Finally, the *Selection* of the individuals that will form the next generation ($g + 1$) is defined in (3), where $f(\cdot)$ is

Algorithm 1: Differential Evolution

Require: NP, L, F, C_r – control parameters
Initialize $\mathbb{P}^{g=0} \leftarrow \{ind_1, \ldots, ind_{NP}\}$
Evaluate $f(\mathbb{P}^g) \leftarrow \{f(ind_1), \ldots, f(ind_{NP})\}$
while (stopping criteria are not met) **do**
  for each $ind \in \mathbb{P}^g$ **do**
    $\mathbb{P}^g \rightarrow \{\varepsilon_1, \varepsilon_2, \varepsilon_3\}$
    $\omega \leftarrow Mutation\,(\varepsilon_1, \varepsilon_2, \varepsilon_3, F)$
    $\gamma \leftarrow Crossover\,(\omega, ind, C_r)$
    $S_{ind} \leftarrow Selection\,(\gamma, ind)$
  end
  $g \leftarrow g + 1$
  $Evaluate\, f(\mathbb{P}^g) \leftarrow \{f(ind_1), \ldots, f(ind_{NP})\}$
end
Return: $\mathbb{P}^g$

a fitness function.

$$S_{ind} = \begin{cases} \gamma & \text{if } f(\gamma) < f(ind) \\ ind_j & \text{otherwise} \end{cases} \tag{3}$$

The process is repeated until one or more stopping criteria are satisfied. A pseudocode of DE is shown in **Algorithm**1.

Once the evolutionary operators are applied, the resulting vectors may contain non-integer values. These are rounded and clamped within the adequate range (*cf.* [45]) before carrying out the Mapping Process, described below.

D. DEPTH-FIRST MAPPING PROCESS

A mapping process allows us to make a distinction between the search space and the solution space; this is an important aspect because it explains the use of different Search Engines that operate only in the search space [46]. In this sense, the candidate descriptors, referred to as 'individuals' in their genotypic-form are evolved in an unconstrained evolutionary search, without knowledge of their phenotypic equivalent but knowing only their fitness values [52]. Different mappings have been reported in the literature, such as Depth-First (the canonical mapping in grammatical evolution), Breadth-First and Position Independent Grammatical Evolution (πGE). In this work, Depth-First is adopted because it is able to generate larger tree structures more quickly than the other mappings [46], implying a lower computational cost. In addition, it has been shown that Depth-First performs better than Breadth-First and similarly to πGE [46].

The Depth-First mapping process takes the genotypic-form of an individual and the CFG-BNF to construct a derivation tree by expanding all the left-most Non-terminal symbols with a production rule selected by (4) [25],

$$Rule = C_v \% N_r \tag{4}$$

where C_v is a codon value which belongs to the individual, N_r is the number of production rules available for the current Non-terminal and % represents the modulus operator, which ensures that C_v is mapped into the interval $[0, N_r - 1]$. The process is repeated until neither Non-terminal symbols, nor codons, remain in the genotypic-form of an individual.

An example of the construction of a texture descriptor is shown in Appendix. Note that during the evolutionary process it is possible to generate texture descriptors that still contain Non-terminal symbols, due to an insufficient number of codons to expand them. In that case, those descriptors are considered invalid and are discarded. An example of this situation is also shown in Appendix.

E. FITNESS FUNCTION

Once the phenotypic-form of a candidate descriptor has been obtained, it is necessary to quantify its quality, i.e. how well it performs on the classification problem at hand. To achieve this, a knowledge database is created from an image dataset through a Feature Extraction Process that employs the candidate descriptor (Fig. 2).

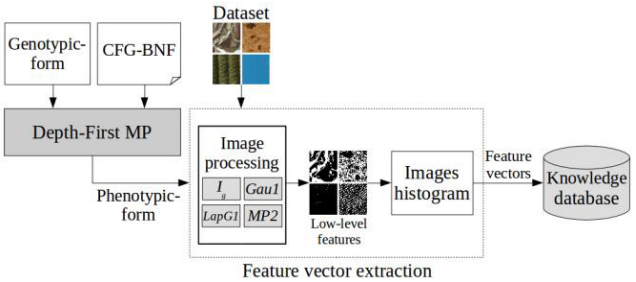


FIGURE 2. Feature extraction process.

In the feature extraction process, a candidate descriptor is applied to all images of a texture dataset in order to extract their low-level features. Then, a histogram-based feature vector $h(v)$ is calculated for each image according to (5), where $I(i, j)$ is an 8-bit (most common color-depth) texture image of size $m \times n$ pixels, f_d is a candidate texture descriptor, $v \in \{0, 1, \ldots, 2^8 - 1\}$, and $\delta(0) = 1$ and zero elsewhere.

$$h(v) = \sum_{i=1}^{m} \sum_{j=1}^{n} \delta\left(f_d\left(I\left(i, j\right)\right) - v\right) \tag{5}$$

After this, all histogram-based feature vectors are labeled according to their texture class and recorded in a knowledge database.

An example of the application of a generated texture descriptor to a texture image is illustrated in Fig. 3. The texture descriptor is $MP2((Ig/Gau1(LapG2(Ig))))$, and the operations applied step by step are listed in Fig. 3, top to bottom and then left to right. In the final step (labelled "6.-Histogram"), a histogram-based feature vector is computed using (5).

The knowledge database is then classified under a cross-validation scheme and an average classification error-rate is computed, which becomes the fitness value of the candidate texture descriptor. The fitness value is obtained using (6), where $E_r \in [0, 1]$ is the classification error rate, obtained

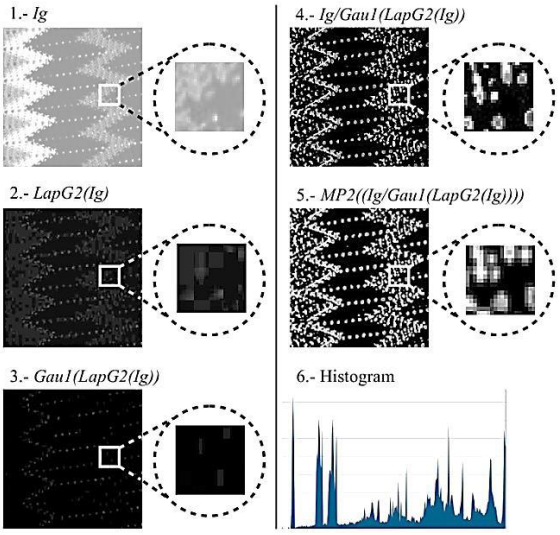


FIGURE 3. Example of operations applied step-by-step. The contrast of the images in this figure has been exaggerated for clarity.

by subtracting from 1 the average of the accuracies (acc_i) of a given classifier using K-folds cross-validation [27].

$$E_r = 1 - \left(\frac{\sum_{i=1}^{K} Classifier\,[acc_i]}{K} \right) \tag{6}$$

This concludes the explanation of the process for computing the fitness value of a candidate texture descripto.

III. EXPERIMENTAL SETUP

This section describes the experimental setup followed in this work to test the effectiveness of our proposed method, including the texture datasets and parameter settings.

A. DATASETS

Five well-known texture datasets [35] used in the literature are employed. A concise description of these datasets is provided below. Some examples of the texture datasets used in this work are shown in Fig. 4.

The *Kylberg texture* dataset contains 28 classes of texturized surfaces, such as fabrics, grains, sesame seeds, stone, lentils, etc. as images of size 576×576 pixels. Despite its large number of classes, it is one of the simplest datasets to classify because as distortions it only includes rotations.

The *UMD texture* dataset comprises 25 classes of textures such as fruits, plants, shelves of bottles, floors, buckets, etc. in high- resolution images (1280×960 pixels) with distortions including differences in illumination and viewpoint.

The *UIUC* dataset includes 25 classes of textures such as gravel, wood, marble, fur, etc. as well as mixtures of them in images of resolution 640×480 pixels. This dataset is probably the most challenging one because the images show several different distortions, such as rotations, viewpoint changes, and differences in scale and illumination.

The *KTH-TIPS* dataset comprises 10 classes of different texturized materials: sandpaper, crumpled aluminum foil, linen, sponge, Styrofoam, cotton, corduroy, brown bread, crackers and orange peel, in images of size 200×200 pixels with changes in rotations, scale, and illumination.

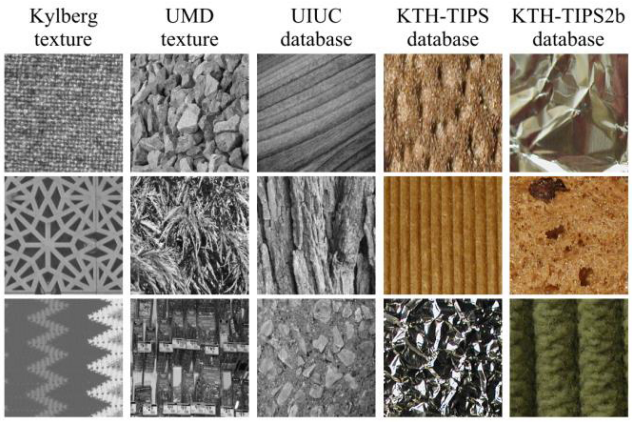


FIGURE 4. Image examples from the datasets Used in this work.

The *KTH-TIPS2b* dataset is an extension of *KTH-TIPS*. In the literature there are two versions of this dataset, one of them with only 11 classes and another one with 44 classes. In this work the latter version is used. Like KTH-TIPS, the KTH-TIPS2b dataset contains different samples of materials with changes in rotations, scale, and illumination.

In our experiments, all the images in the different datasets were transformed into gray-level images and resized to 512×512 pixels (the average of their original resolutions) in order to use a standard resolution for processing them.

State-of-the-art results of Deep Learning and Evolutionary approaches for classification of the datasets used in this work are shown in Table 3. Said results constitute an overview of the current techniques developed to solve the problem of texture classification.

B. EXPERIMENTS

Experiments were performed to assess the quality of the descriptors generated by means of our methodology. A single experiment considers one texture dataset of the ones listed above and one the following classifiers: Support Vector Machine (SVM), k-Nearest Neighbor (k-NN) and Naïve Bayes (NB). To ensure the consistency of the experimental results, each experiment is executed 31 times independently and the best texture descriptor found each time is recorded. Finally, the median classification performance and standard deviation of the 31 best texture descriptors are reported as the results for a particular pair of dataset and classifier.

To compare the effectiveness of our proposed method, the texture descriptor corresponding to the median of the 31 experiments described above is measured against six hand-crafted texture descriptors, namely: Histogram of Oriented Gradients (HOG) [24], Haralick's Texture Features (HTF) [33], Multifractal Spectrum (MFS) [28], Daubechies-4 wavelet transform (Daub-4) [1], [56], Local Binary Patterns (LBP) [39] and Gabor Filter Banks (GFB)

TABLE 3. Classification results (%) of different state-of-the-art approaches (Mean ± Std. Dev[a]).

Dataset:	Kylberg	UMD	UIUC	KTH-TIPS	KTH-TIPS2[b]
Deep Learning Approaches					
ScaNet (PCA) [18]	–	98.4	96.15	–	68.92
ScaNet (NNC) [18]	–	93.36	88.64	–	63.66
PCANet (NNC) [20]	–	90.5	57.7	–	59.43
RandNet (NNC) [20]	–	90.87	56.57	–	60.67
VGG-VD (FV_CNN) [22]	–	–	–	–	81.8 ± 2.5
CNN [11]	–	–	49.75	34.93	40.29
T-CNN-3 [7]	99.4 ± 0.2	–	–	–	73.2 ± 2.2
AlexNet [7]	99.4 ± 0.1	–	–	–	71.5 ± 1.4
SoA [7]	99.7	–	–	–	81.5 ± 2.0
TEX-Net-LF [9]	–	–	–	–	75.5 ± 2.7
CNN 5C [12]	–	–	91.18	69.5	68.67
CDBN 2 layer [12]	–	–	70.02	59.15	82.24
Evolutionary Approaches					
GP-criptor (1NN) [5]	93.21 ± 1.14	–	–	–	–
TLGP-criptor (1NN) [37]	87.26 ± 0.48	–	–	–	–
GP-criptor (1NN) [2]	88.5 ± 1.4	–	–	–	–
GP-criptor (NB) [2]	72.7 ± 3.8	–	–	–	–
GP-criptor (SVM) [2]	68.3 ± 1.8	–	–	–	–
$MGPD^{ri}_{9,5}$ (1NN) [3]	90.91	–	–	–	–
EID^{ri} (1NN) [4]	88.6	–	–	–	–
GP-criptor^{ri} (1NN) [36]	88.5 ± 1.4	–	–	–	–

[a]Some studies did not report a standard deviation. [b]This version of the dataset included only 11 classes.

[16]. In this comparison, the classification parameters are kept constant and 10-fold cross-validation is used to obtain the accuracy of each hand-crafted descriptor on each of the texture datasets for each of the classifiers listed above.

C. PARAMETER SETTINGS

The configuration of the different algorithms involved in the generation of problem-adapted texture descriptors is reported in this section, including those parameters used in their evolution and for the classifiers used in their evaluation.

1) EVOLUTIONARY PARAMETERS

As mentioned in Section II, during the evolutionary process a population of individuals is transformed into candidate texture descriptors that are evaluated on a texture dataset. Since image processing can be very expensive, the number of individuals used in the experiments is relatively small, to limit the computational cost. The population's size is $N_P = 100$; and each individual is formed by $L = 100$ codons with values in the range [0→255]. The constant of differentiation is $F = 0.5$ and the crossover rate is $Cr = 0.8$; these values are suggested in [25]. The evolutionary process ends if an individual with a Fitness Value $= 0$ is found, or if the Maximum Number of Generations $= 500$ is reached.

2) CLASSIFIERS PARAMETERS

Since the proposed methodology creates a problem-adapted texture descriptor guided by the performance of a specific classifier, SVMs, k-NN and NB are used in order to show that, independently of the classifier, the method is capable of evolutionarily designing texture descriptors that can compete with the performance of hand-crafted ones. The classifiers are obtained from the Waikato Environment for Knowledge Analysis (WEKA) software [32].

SVM hyper-parameters are usually determined in practice by Grid Search [14], evolutionary algorithms [30] or combining both techniques [41]. However, parameter tuning is outside the scope of this work, and performing it for each individual candidate descriptor generated during the evolutionary process could become prohibitively expensive. For these reasons, SVMs with fixed parameters are used.

Two SVMs, one with Linear Kernel and one with a Radial Basis Function (RBF) Kernel are used with their default parameters in WEKA ($C = 1$ and $\gamma = 0.01$); small values of C and γ tend to produce smooth decision surfaces and prevent overfitting [6]. The k-NN algorithm is tested with $k = 1, 3$. The NB classifier is used with a Normal probability distribution. Cross-validation is applied with $K = 10$ folds.

IV. RESULTS AND ANALYSIS

In this section, the results of the experiments are presented and discussed, starting with the results of the evolutionary process. The median and standard deviation of the best (under 10-fold cross-validation) fitness values over 31 independent runs are shown in Table 4. The problem-adapted descriptors whose performance is the median of the experiments are also reported.

Notice that for all the experiments the order of the standard deviation is between 1E-4 and 1E-2; this shows a high consistency in the descriptors designed by our method, in the sense that their performance is very similar. With regards to the operators of the problem-adapted image descriptors, it can be seen that most are formed by the combination of low-pass and high-pass filters that interact with each other via arith-

TABLE 4. Median Fitness over 31 runs Using a Particular Classifier (Median ± Std. Dev)

Classifier	Dataset	Fitness	Problem-adapted Image Descriptor (PAID)
Linear SVM	Kylberg	0.0011 ± 0.0180	MP2((Ig/Gau1(LapG2(Ig))))
	UMD	0.0360 ± 0.0082	MP2((Ig/LapG2(Lap(Ig))))
	UIUC	0.1270 ± 0.0229	MP2((Ig/MedF(LapG2(Ig))))
	KTH-TIPS	0.1395 ± 0.0049	MP2(Lap((T0.5((Ig/LapG2(Lg(Ig))))+MedF(Ig))))
	KTH-TIPS2b	0.3895 ± 0.0134	MP2(LapG1(Gau1(Ig)))
Radial SVM	Kylberg	0.0971 ± 0.0094	MP2(LapG2(Ig))
	UMD	0.2370 ± 0.0345	MP2(LapG2(AverF(Ig)))
	UIUC	0.3830 ± 0.0119	MP2(LapG2(Gau1(Ig)))
	KTH-TIPS	0.4370 ± 0.0249	MP2(Sqrt(AbsV(LapG2((LapG1((Ig-a(Ig + (Ig * MedF(GFB135(GFB45(Ig))))))) - (Gau1(Sqrt(AbsV(Ig))) +Ig))))))
	KTH-TIPS2b	0.5983 ± 0.0007	MP2(LapG2(Gau2(Ig)))
1-NN	Kylberg	0.0270 ± 0.0057	MP2(LapG2(Ig))
	UMD	0.0360 ± 0.0247	MP2(Sqrt((Ig/LapG2(Lap(Ig)))))
	UIUC	0.2150 ± 0.0340	MP2((Ig/Gau1(LapG2((Ig+Ig)))))
	KTH-TIPS	0.1012 ± 0.0468	MP2((LapG2(Ig)+(GFB0(Ig)/LapG2(Ig))))
	KTH-TIPS2b	0.3504 ± 0.0258	MP2(Lap(LapG1(Ig)))
3-NN	Kylberg	0.0129 ± 0.0036	MP2((Ig/MedF(LapG2(Ig))))
	UMD	0.0670 ± 0.0076	MP2((Ig/MedF(LapG2(Lap(Ig)))))
	UIUC	0.2120 ± 0.0236	MP2((Ig/LapG1(LapG2(Ig))))
	KTH-TIPS	0.2111 ± 0.0132	MP2(LapG1(Gau1(Ig)))
	KTH-TIPS2b	0.3533 ± 0.0180	MP2(Lap(LapG1(Ig)))
Naïve Bayes	Kylberg	0.0617 ± 0.0188	MP2(LapG1(Gau1(Ig)))
	UMD	0.1230 ± 0.0076	MP2(LapG1(AverF(Ig)))
	UIUC	0.2490 ± 0.0009	MP2(LapG2(Gau1(Ig)))
	KTH-TIPS	0.3728 ± 0.0417	MP2(LapG1((((Ig*Ig)*Ig)-a(GFB90(Ig)))))
	KTH-TIPS2b	0.5486 ± 0.0017	MP2(LapG1(Gau1(Ig)))

metic operations, especially divisions. In every case, once the features have been extracted, the Max-pooling operation is applied to reduce the size of the representation, preserving the most relevant information. In every instance the evolutionary process preferred a window of size 2×2 for the Max-pooling operation.

Interestingly, for the KTH-TIPS dataset, the problem-adapted descriptors were the largest (according to the number of primitive operators that these contain) regardless of the classifier used in their evaluation. Additionally, in most cases the evolutionary process led the design towards the use of Gabor Filter Banks.

The performance (classification accuracy) of our Problem-Adapted Image Descriptors (hereafter referred to as PAID for brevity) is compared against that of hand-crafted texture descriptors in Table 5 for each of the considered classifiers (Linear SVM, Radial SVM, 1-NN, 3-NN and NB respectively). The values in bold typeface represent the best classification accuracies on each dataset. The results in Table 5 are summarized below.

Using Linear SVM: on the Kylberg dataset the PAID methodology achieves a performance close to 100%, outperforming all the other descriptors; on the UMD dataset PAID is placed second, with similar performance to that of LBP; on the UIUC database, PAID significantly outperforms all other descriptors; on KTH-TIPS dataset, only LBP performs slightly better than PAID; finally, on KTH-TIPS2b, PAID outperforms all other descriptors except for LBP.

Using Radial SVM: PAID achieves higher performance than all other descriptors on all datasets except for KTH-TIPS. However, its performance this time is lower than that achieved using Linear SVM.

Using 1-NN: PAID, MFS and LBP obtain the top 3 results on all datasets, with very similar performance relative to each other. The only exception is on the KTH-TIPS2b dataset, where GFB performs significantly better than all others.

Using 3-NN: on the Kylberg dataset PAID obtains the highest performance, closely followed by LBP; on the UMD and UIUC datasets the highest performance is obtained by MFS, closely followed by PAID; on KTH-TIPS the highest performance is obtained by MFS, followed by LBP; on KTH-TIPS2b the GFB is highest, and followed by LBP.

Using Naïve Bayes: PAID performs better than the other descriptors on all datasets except for KTH-TIPS2b, where the LBP descriptor outperforms all other descriptors.

A statistical analysis based on non-parametric statistical tests is performed in order to provide statistical support to the results shown in Table 5. This analysis is described below. Notice that performing a similar comparison against the results reproduced in Table 3 would be inadequate, since those results come from a diversity of works that follow different experimental methodologies and experimental setups; however, said results provide a general context against which one can place and visualize the potential of our proposed methodology.

A. STATISTICAL ANALYSIS OF THE RESULTS

Three non-parametric tests are conducted to determine the existence of statistically significant differences between the performances of the texture descriptors compared, based on

TABLE 5. Classification accuracies (%) of seven texture descriptors (Median ± Std. Dev).

Classifier	Dataset	HOG	HTF	MFS	Daub-4	GFB	LBP	Proposed (PAID)
Linear SVM	Kylberg	48.97 ± 1.45	76.96 ± 1.83	97.79 ± 0.43	95.15 ± 1.04	89.68 ± 1.69	97.81 ± 0.21	**99.22 ± 0.31**
	UMD	51.40 ± 6.34	76.30 ± 5.39	94.30 ± 3.58	74.20 ± 2.92	70.60 ± 2.45	**97.50 ± 1.26**	97.40 ± 1.90
	UIUC	24.60 ± 4.05	55.90 ± 4.23	84.20 ± 2.89	50.10 ± 5.28	50.20 ± 5.23	78.10 ± 5.05	**87.30 ± 3.76**
	KTH-TIPS	62.72 ± 6.87	58.52 ± 5.90	81.60 ± 3.00	49.25 ± 4.22	71.72 ± 4.69	**89.01 ± 3.16**	86.05 ± 4.13
	KTH-TIPS2b	43.79 ± 1.27	40.45 ± 2.48	60.16 ± 1.96	36.42 ± 2.50	50.20 ± 5.23	**80.57 ± 1.87**	64.58 ± 1.98
Radial SVM	Kylberg	33.37 ± 2.30	56.36 ± 2.28	78.44 ± 2.13	71.81 ± 3.22	64.35 ± 2.20	89.39 ± 1.46	**90.29 ± 0.74**
	UMD	29.50 ± 4.36	57.70 ± 4.78	64.70 ± 3.16	46.10 ± 4.39	49.90 ± 3.70	74.10 ± 4.03	**76.30 ± 3.92**
	UIUC	13.40 ± 2.58	36.40 ± 4.22	43.10 ± 3.05	31.70 ± 3.63	25.70 ± 4.47	38.50 ± 2.10	**61.70 ± 2.10**
	KTH-TIPS	22.72 ± 5.47	41.48 ± 5.64	53.33 ± 4.20	34.56 ± 3.70	39.51 ± 2.65	**59.75 ± 7.14**	56.30 ± 3.18
	KTH-TIPS2b	13.70 ± 1.05	13.51 ± 0.96	23.34 ± 2.53	16.04 ± 1.56	22.71 ± 2.06	39.35 ± 3.50	**40.17 ± 2.06**
1-NN	Kylberg	57.86 ± 1.54	93.30 ± 1.02	96.16 ± 0.77	84.56 ± 1.33	93.35 ± 1.51	96.97 ± 0.36	**97.30 ± 0.53**
	UMD	72.00 ± 3.89	90.80 ± 2.86	95.90 ± 1.51	71.40 ± 2.94	82.90 ± 2.34	95.76 ± 1.74	**96.40 ± 1.56**
	UIUC	33.80 ± 3.52	65.30 ± 3.52	**82.30 ± 1.72**	43.70 ± 5.68	59.90 ± 3.81	72.40 ± 2.58	78.50 ± 2.11
	KTH-TIPS	79.01 ± 4.73	75.93 ± 4.59	**90.62 ± 2.72**	67.41 ± 4.69	88.15 ± 2.83	86.05 ± 3.46	89.88 ± 4.69
	KTH-TIPS2b	63.70 ± 2.34	57.66 ± 2.64	67.02 ± 2.30	42.45 ± 2.45	**71.46 ± 2.32**	67.63 ± 1.65	64.96 ± 1.06
3-NN	Kylberg	55.87 ± 1.03	93.13 ± 0.81	96.07 ± 0.70	85.40 ± 1.75	92.95 ± 1.15	98.33 ± 0.44	**98.71 ± 0.61**
	UMD	65.60 ± 3.80	89.20 ± 2.36	**95.30 ± 2.05**	70.20 ± 3.05	78.20 ± 2.82	95.30 ± 2.48	93.30 ± 2.00
	UIUC	28.40 ± 5.64	64.90 ± 4.50	**81.00 ± 3.30**	42.00 ± 5.35	55.20 ± 4.04	66.30 ± 5.34	78.80 ± 4.15
	KTH-TIPS	74.57 ± 5.75	77.04 ± 3.88	**87.53 ± 2.92**	65.68 ± 5.51	83.09 ± 2.33	84.57 ± 4.03	78.89 ± 4.87
	KTH-TIPS2b	57.53 ± 1.56	56.25 ± 2.56	65.85 ± 2.85	42.53 ± 2.08	**68.03 ± 2.25**	67.70 ± 1.68	64.67 ± 2.67
Naïve Bayes	Kylberg	38.53 ± 1.66	80.60 ± 1.72	86.63 ± 1.06	87.46 ± 1.58	79.78 ± 1.77	91.27 ± 1.60	**93.83 ± 0.73**
	UMD	50.00 ± 4.29	80.40 ± 4.20	84.70 ± 2.61	63.80 ± 2.23	60.90 ± 2.26	84.20 ± 5.06	**87.70 ± 4.45**
	UIUC	20.70 ± 2.87	58.90 ± 4.18	63.20 ± 5.44	39.00 ± 3.69	40.80 ± 4.19	56.60 ± 3.23	**75.10 ± 3.65**
	KTH-TIPS	42.96 ± 6.31	55.56 ± 5.44	60.25 ± 4.55	33.70 ± 3.91	50.37 ± 3.48	62.71 ± 5.81	**62.72 ± 4.78**
	KTH-TIPS2b	27.29 ± 1.58	41.60 ± 3.41	34.55 ± 1.40	20.96 ± 2.02	35.92 ± 2.20	**57.83 ± 1.58**	45.14 ± 2.09

TABLE 6. Ranks of texture descriptors according to Friedman (F), Aligned Friedman (AF), and Quade (Q) Tests.

	Linear SVM			Radial SVM			1-NN			3-NN			Naïve Bayes		
Descriptors	F	AF	Q	F	AF	Q	F	AF	Q	F	AF	Q	F	AF	Q
PAID	**1.6**	7.8	**1.4**	**1.2**	**4.4**	**1.1**	**2**	9.4	**2**	2.8	11	2.3	**1.2**	**6.1**	**1.1**
LBP	**1.6**	**5.6**	1.9	2	7.4	2.1	2.8	10.4	2.6	2.1	9.1	2.2	2.4	8.8	2.6
MFS	2.8	10.8	2.7	2.8	12.8	2.7	**2**	**8.4**	2.1	**1.9**	**7.3**	**1.9**	3.2	12.8	3.1
HT	5.2	24.8	4.9	5	22.6	4.9	5	22	4.8	4.6	19	4.3	3.8	16.6	3.9
GFB	4.8	21.2	5	5	23	5.2	3.6	16.6	3.8	3.8	20.2	4.3	5.2	24.6	5.4
Daub-4	5.8	25.6	5.5	5.2	23.4	5	6.6	30.4	6.4	6.4	29.2	6.2	5.6	25.8	5
HOG	6.2	30.2	6.6	6.8	32.4	6.9	6	28.8	6.3	6.4	30.2	6.7	6.6	31.2	6.8

the results reported in Table 5. The non-parametric tests are: Friedman (F), Aligned Friedman Ranks (AF), and Quade (Q); these tests are commonly used when the assumption of Normality in the data distributions cannot be guaranteed and the sample size is small [26]. The tests are obtained from the CONTROLTEST and the MULTIPLETEST packages [26].

The ranks computed by means of each non-parametric test are reported in Table 6. Notice that the smallest rank corresponds to the best-performing descriptor and so on. The numbers in bold typeface indicate the best ranked descriptor per column. Although the rank order varies depending on the non-parametric test and the classifier, it can be observed that PAID is consistently (in 4 out of 5 cases) ranked first under the Quade test (which provides a more robust statistic than the other tests because it takes into account the difficulty of the problems [26]). PAID is also consistently (again in 4 out of 5 cases) ranked first under the Friedman test. These results support the consistency of our method.

The p-value for each non-parametric test is reported in Table 7. These values indicate whether the hypothesis test is statistically significant or not at confidence $\alpha = 0.05$. The null hypothesis states that all the data samples come from distributions with equal means; p-values lower than the confidence level support the alternative hypothesis: that one or more of the correlated samples is statistically different. From Table 7 it can be appreciated that there are statistically significant differences; therefore, post-hoc tests for multiple comparisons are conducted in order to discern which of the pairs of compared texture descriptors show significant differences between them.

TABLE 7. p-values of non parametric tests.

Classifier	Friedman	Aligned Friedman	Quade
Linear SVM	3.76E-4	6.33E-1	3.74E-5
Radial SVM	1.95E-4	6.34E-1	6.64E-6
1-NN	8.36E-4	6.33E-1	2.55E-4
3-NN	1.70E-6	6.32E-1	1.13E-4
Naïve Bayes	7.43E-7	6.33E-1	2.24E-4

The adjusted p-values computed through Holm's procedure are reported in Table 8 for all possible pairs of texture descriptors and grouped column-wise according to the

TABLE 8. Adjusted p-Values according to Holm's procedure for all pairs of descriptors.

Pairs of Descriptors	Classifiers				
	Linear SVM	Radial SVM	1-NN	3-NN	Naïve Bayes
PAID vs HOG	**2.00E-6**	**9.07E-11**	**5.03E-4**	**1.45E-3**	**3.00E-6**
PAID vs HTF	**8.10E-5**	**1.56E-7**	**1.06E-2**	2.68E-1	**2.12E-2**
PAID vs MFS	4.98E-1	**9.87E-3**	1	1	1.28E-1
PAID vs Daub-4	**8.00E-6**	**6.73E-8**	**8.00E-5**	**1.45E-3**	**6.70E-5**
PAID vs GFB	**3.39E-4**	**1.56E-7**	4.92E-1	1	**2.34E-4**
PAID vs LBP	1	4.19E-1	1	1	5.72E-1
LBP vs HOG	**2.00E-6**	**2.05E-9**	**5.88E-3**	**1.60E-4**	**1.25E-4**
LBP vs HTF	**8.10E-5**	**7.51E-6**	9.76E-2	**3.84E-2**	5.44E-1
LBP vs MFS	4.98E-1	4.19E-1	1	1	8.57E-1
LBP vs Daub-4	**8.00E-6**	**2.80E-6**	**8.67E-4**	**1.60E-4**	**3.31E-3**
LBP vs GFB	**3.39E-4**	**7.51E-6**	1	2.68E-1	**1.24E-2**
GFB vs HOG	2.85E-1	**4.09E-3**	5.77E-2	**3.25E-2**	5.44E-1
GFB vs HTF	1	1	5.77E-1	1	5.44E-1
GFB vs MFS	**3.33E-2**	**5.05E-4**	4.92E-1	2.21E-1	1.28E-1
GFB vs Daub-4	6.98E-1	1	**1.06E-2**	**3.25E-2**	8.57E-1
Daub-4 vs HOG	1	**9.87E-3**	1	1	7.40E-1
Daub-4 vs HTF	1	1	4.92E-1	2.68E-1	1.95E-1
Daub-4 vs MFS	**6.62E-4**	**1.76E-4**	**8.00E-5**	**9.20E-5**	**3.83E-2**
MFS vs HOG	**1.64E-1**	**6.73E-8**	**5.04E-4**	**9.20E-5**	**1.76E-3**
MFS vs HTF	**7.27E-3**	**5.05E-4**	**1.06E-2**	**2.52E-2**	8.57E-1
HTF vs HOG	6.98E-1	**4.09E-3**	1	2.68E-1	1.24E-1

specific classifier used. The values in bold typeface indicate that there is enough statistical evidence to reject the null hypothesis at confidence level $\alpha = 0.05$. Considering these results and the ranks in Table 6, it can be concluded that our problem-adapted image descriptors are quite competitive against the other descriptors considered, overcoming most of them (HOG, HTF, Daub-4 and GFB) with statistically significant differences and achieving equivalent performance to those of LBP and MFS.

V. CONCLUSION

In this paper we proposed a GE approach to deal with texture classification problems in a general way, by automatically designing problem-adapted image descriptors capable of detecting and extracting relevant features from images. The extracted features provide an effective representation that favors performance, measured as classification accuracy.

The proposed methodology was tested on five different texture datasets that present images with distortions such as translations, rotations, differences in illumination and scale. Three different classification algorithms (with different parameters) were also employed to drive the evolutionary process and show that our methodology is successful, independently of the classifier. Statistical analysis of the results indicate that our descriptors outperform the HOG, HTF, Daub-4, and GFB texture descriptors, with statistically significant differences and achieve classification accuracies that are on a par with those of MFS and LBP.

Thus, the ability of our method to generate relatively general solutions for feature extraction and representation of texture datasets was demonstrated. More importantly, this is achieved without prior knowledge of the specific problems and without human intervention.

As future work, the robustness of our method will be investigated against more difficult textures datasets, e.g. those involving a larger number of classes, other kinds of image distortions and real-life texture images. Due to the use of a grammar, it is possible to expand our design of the CFG-BNF to easily add domain knowledge; this offers a great possibility to continue experimenting with additional feature detection filters as primitive operators, without having to alter other components of our method. In addition, it would be interesting to perform parameter optimization of the image descriptors automatically designed by our method, expecting improvements in the classification process.

APPENDIX
CONSTRUCTION OF A TEXTURE DESCRIPTOR

A Depth-First mapping process that takes an individual in its genotypic-form and a CFG-BNF are used to construct a derivation tree from which a functional program (a texture descriptor) is subsequently extracted in its phenotypic-form.

To build the derivation tree in Fig. 5, the production rules P (Section II-B) and the individual shown in (7) are required. As mentioned in Section II-A, each value in the individual is a codon value (C_v) and N_r is the number of rules available for the current non-terminal symbol to be expanded.

$$individual = \{233, 20, 88, 3, 78, 34, 67, 20, 35, 57\} \quad (7)$$

In this example, the first C_v is 233, and the initial symbol to be expanded is $\langle start \rangle$, with $N_r = 1$ (see the production rules P). The expansion order is indicated in Fig. 5, by the numbers on the arrows between blocks. For instance: **1** (233%1=0), means that the expansion order is **1** and (233%1=0) is the application of (4).

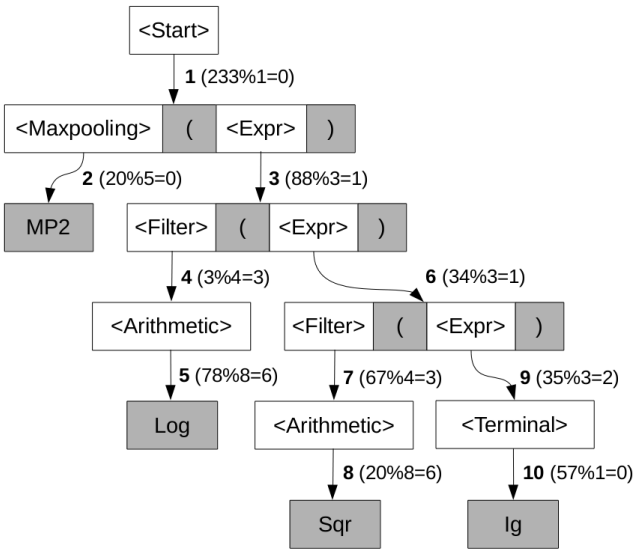


FIGURE 5. Derivation tree of the depth-first mapping process.

In the first expansion, the result of (4) shows that $\langle start \rangle$ must be expanded using the production rule 0, in this case: $\langle Maxpooling \rangle$ ($\langle Expr \rangle$) .

After the first derivation, the second codon value $C_v = 20$ is read and the left-most non-terminal symbol is selected to be expanded, in this example: $\langle Maxpooling \rangle$, which has $N_r = 5$. Applying (4) to expand this non-terminal symbol corresponds to the operation: 20%5=0; this means that $\langle Maxpooling \rangle$ must be expanded using the production rule 0, in this case: $MP2$, which is a terminal symbol.

For the third expansion, $C_v = 88$ and the left-most Non-terminal symbol is $\langle Expr \rangle$ with $N_r = 3$. Performing the operation 88%3=1; thus $\langle Expr \rangle$ is expanded using the production rule 1, i.e. $\langle Filter \rangle$ ($\langle Expr \rangle$) .

The mapping process continues in this manner, always replacing the left-most non-terminal symbol through a production rule available for that symbol in the CFB-BNF and chosen based on a codon value. The process ends when all non-terminal symbols are transformed into elements of the terminal set T.

Finally, the proposed program (8), formed by the terminal symbols (gray squares in Fig. 5), is parsed from the derivation tree, and evaluated via the fitness function.

$$Solution = MP2(Log(Sqr(Ig))) \tag{8}$$

A. INVALID TEXTURE DESCRIPTOR

If all the codon values of an individual are consumed and the program still contains Non-terminal symbols, then the latter is considered invalid. For instance, if the individual: {233, 20, 88, 3, 78, 34, 67, 20, 55, 57} was used to build a derivation tree, the generated program (9) would still contain the non-terminal symbols $\langle \boldsymbol{lap} \rangle$ and $\langle \boldsymbol{Expr} \rangle$; therefore, this solution could not be evaluated by the fitness function.

$$Solution = MP2(Log(Sqr(\langle lap \rangle\ (\langle Expr \rangle)))) \tag{9}$$

VI. ACKNOWLEDGMENT

The authors want to thank to the National Technological Institute of Mexico for the support provided for this research.

REFERENCES

[1] P. S. Addison, *The Illustrated Wavelet Transform Handbook: Introductory Theory and Applications in Science, Engineering, Medicine and Finance*. Boca Raton, FL, USA: CRC Press, 2017.
[2] H. Al-Sahaf, A. Al-Sahaf, B. Xue, M. Johnston, and M. Zhang, "Automatically evolving rotation-invariant texture image descriptors by genetic programming," *IEEE Trans. Evol. Comput.*, vol. 21, no. 1, pp. 83–101, Feb. 2017.
[3] H. Al-Sahaf, B. Xue, and M. Zhang, "A multitree genetic programming representation for automatically evolving texture image descriptors," in *Proc. Asia–Pacific Conf. Simulated Evol. Learn.*, 2017, pp. 499–511.
[4] H. Al-Sahaf, M. Zhang, A. Al-Sahaf, and M. Johnston, "Keypoints detection and feature extraction: A dynamic genetic programming approach for evolving rotation-invariant texture image descriptors," *IEEE Trans. Evol. Comput.*, vol. 21, no. 6, pp. 825–844, Dec. 2017.
[5] H. Al-Sahaf, M. Zhang, M. Johnston, and B. Verma, "Image descriptor: A genetic programming approach to multiclass texture classification," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, May 2015, pp. 2460–2467.
[6] E. Alpaydin, *Introduction to Machine Learning* (Adaptive Computation and Machine Learning). Cambridge, MA, USA: MIT Press, 2004.
[7] V. Andrearczyk and P. F. Whelan, "Using filter banks in convolutional neural networks for texture classification," *Pattern Recognit. Lett.*, vol. 84, pp. 63–69, Dec. 2016.
[8] H. C. Andrews, *Introduction to Mathematical Techniques in Pattern Recognition*. Hoboken, NJ, USA: Wiley, 1972.
[9] R. M. Anwer, F. S. Khan, J. van de Weijer, M. Molinier, and J. Laaksonen. (2017). "Binary patterns encoded convolutional neural networks for texture recognition and remote sensing scene classification." [Online]. Available: https://arxiv.org/abs/1706.01171
[10] J. Bala, K. De Jong, J. Huang, H. Vafaie, and H. Wechsler, "Using learning to facilitate the evolution of features for recognizing visual concepts," *Evol. Comput.*, vol. 4, no. 3, pp. 297–311, 1996.
[11] S. Basu *et al.*, "A theoretical analysis of deep neural networks for texture classification," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 992–999.
[12] S. Basu, S. Mukhopadhyay, M. Karki, R. Di Biano, S. Ganguly, R. Nemani, and S. Gayaka, "Deep neural networks for texture classification—A theoretical analysis," *Neural Netw.*, vol. 97, pp. 173–182, Jan. 2018.
[13] S. Basu, H. Nanda, K. Sinha, and A. Raja, "Deep neural networks for texture classification," *Neural Netw. & Mach. Learn.*, vol. 1, no. 1, p. 3, 2016.
[14] A. Ben-Hur and J. Weston, "A user's guide to support vector machines," in *Data Mining Techniques for the Life Sciences*. New York, NY, USA: Springer, 2010, pp. 223–239.
[15] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[16] F. Bianconi and A. Fernández, "Evaluation of the effects of Gabor filter parameters on texture classification," *Pattern Recognit.*, vol. 40, no. 12, pp. 3325–3335, 2007.
[17] Y.-L. Boureau, J. Ponce, and Y. Le Cun, "A theoretical analysis of feature pooling in visual recognition," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 111–118.
[18] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, Aug. 2013.
[19] P. Cavalin and L. S. Oliveira, "A review of texture classification methods and databases," in *Proc. SIBGRAPI-T*, Oct. 2017, pp. 1–8.
[20] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "PCANet: A simple deep learning baseline for image classification," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5017–5032, Dec. 2015.
[21] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support vector machines for histogram-based image classification," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1055–1064, Sep. 1999.
[22] M. Cimpoi, S. Maji, and A. Vedaldi, "Deep filter banks for texture recognition and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3828–3836.

[23] E. Clemente, G. Olague, D. Hernández, J. L. Briseño, and J. Mercado, "Object detection in natural images using the brain programming paradigm with a multi-objective approach," in *Applications of Evolutionary Computation. EvoApplications* (Lecture Notes in Computer Science), vol. 9028, A. Mora and G. Squillero, Eds. Cham, Switzerland: Springer, 2015.

[24] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, vol. 1, no. 1, pp. 886–893.

[25] I. Dempsey, M. O'Neill, and A. Brabazon, *Foundations in Grammatical Evolution for Dynamic Environments*, vol. 194. Berlin, Germany: Springer, 2009.

[26] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011.

[27] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Hoboken, NJ, USA: Wiley, 2012.

[28] C. Fermuller, Y. Xu, and H. Ji, "A view-point invariant texture descriptor," *J. Vis.*, vol. 8, no. 6, p. 354, 2008.

[29] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems," *J. Mach. Learn. Res*, vol. 15, no. 1, pp. 3133–3181, 2014.

[30] F. Friedrichs and C. Igel, "Evolutionary tuning of multiple SVM parameters," *Neurocomputing*, vol. 64, pp. 107–117, Mar. 2005.

[31] I. G. Fellow, Y. Bengio, and A. Courville, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.

[32] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explor. Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.

[33] R. M. Haralick and K. Shanmugam, "Textural features for image classification," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. SMC-6, no. 6, pp. 610–621, Nov. 1973.

[34] B. Hegerty, C.-C. Hung, and K. Kasprak, "A comparative study on differential evolution and genetic algorithms for some combinatorial problems," in *Proc. 8th Mexican Int. Conf. Artif. Intell.*, 2009, pp. 9–13.

[35] S. Hossain and S. Serikawa, "Texture databases-a comprehensive survey," *Pattern Recognit. Lett.*, vol. 34, no. 15, pp. 2007–2022, 2013.

[36] M. Iqbal, B. Xue, H. Al-Sahaf, and M. Zhang, "Cross-domain reuse of extracted knowledge in genetic programming for image classification," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 569–587, Aug. 2017.

[37] M. Iqbal, B. Xue, and M. Zhang, "Reusing extracted knowledge in genetic programming to solve complex texture image classification problems," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*, 2016, pp. 117–129.

[38] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, vol. 1. Cambridge, MA, USA: MIT Press, 1992.

[39] L. Liu, P. Fieguth, Y. Guo, X. Wang, and M. Pietikäinen, "Local binary features for texture classification: Taxonomy and experimental study," *Pattern Recognit.*, vol. 62, pp. 135–160, Feb. 2017.

[40] L. Liu, L. Shao, and X. Li, "Building holistic descriptors for scene recognition: A multi-objective genetic programming approach," in *Proc. 21st ACM Int. Conf. Multimedia*, 2013, pp. 997–1006.

[41] R. Liu, E. Liu, J. Yang, M. Li, and F. Wang, "Optimizing the hyperparameters for SVM by combining evolution strategies with a grid search," in *Intelligent Control and Automation*. Berlin, Germany: Springer, 2006, pp. 712–721.

[42] B. S. Manjunath, P. Salembier, and T. Sikora, *Introduction to MPEG-7: Multimedia Content Description Interface*, vol. 1. Hoboken, NJ, USA: Wiley, 2002.

[43] G. Olague and L. Trujillo, "Evolutionary-computer-assisted design of image operators that detect interest points using genetic programming," *Image Vis. Comput.*, vol. 29, no. 7, pp. 484–498, 2011.

[44] G. Olague and L. Trujillo, "Interest point detection through multiobjective genetic programming," *Appl. Soft Comput.*, vol. 12, no. 8, pp. 2566–2582, 2012.

[45] M. O'Neill and A. Brabazon, "Grammatical differential evolution," in *Proc. IC-AI*, 2006, pp. 231–236.

[46] D. Fagan, M. O'Neill, E. Galván-López, A. Brabazon, and S. McGarraghy, "An analysis of genotype-phenotype maps in grammatical evolution," in *Genetic Programming*. Berlin, Germany: Springer, 2010.

[47] M. O'Neill and A. Brabazon, "Grammatical swarm: The generation of programs by social programming," *Natural Comput.*, vol. 5, no. 4, pp. 443–462, 2006.

[48] P. Pachowicz and J. Bala, "Texture recognition through machine learning and concept optimization," *Rep. Mach. Learn. Inference Lab.*, vol. 1051, pp. 4–95, 1991.

[49] C. B. Perez and G. Olague, "Learning invariant region descriptor operators with genetic programming and the f-measure," in *Proc. 19th Int. Conf. Pattern Recognit. (ICPR)*, 2008, pp. 1–4.

[50] C. B. Perez and G. Olague, "Genetic programming as strategy for learning image descriptor operators," *Intell. Data Anal.*, vol. 17, no. 4, pp. 561–583, 2013.

[51] M. Petrou and P. G. Sevilla, *Image Processing: Dealing With Texture*, vol. 1. Hoboken, NJ, USA: Wiley, 2006.

[52] C. Ryan, J. J. Collins, and M. O. Neill, "Grammatical evolution: Evolving programs for an arbitrary language," in *Proc. Eur. Conf. Genetic Programm.* Berlin, Germany: Springer, 1998, pp. 83–96.

[53] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Artificial Neural Networks—ICANN*. Berlin, Germany: Springer, 2010, pp. 92–101.

[54] L. Shao, L. Liu, and X. Li, "Feature learning for image classification via multiobjective genetic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 7, pp. 1359–1371, Jul. 2014.

[55] L. Shapiro, *Computer Vision and Image Processing*. San Diego, CA, USA: Academic, 1992.

[56] S. Sidhu and K. Raahemifar, "Texture classification using wavelet transform and support vector machines," in *Proc. Can. Conf. Elect. Comput. Eng.*, 2005, pp. 941–944.

[57] P. Sondhi, "Feature construction methods: A survey," Univ. Illinois Urbana–Champaign, Champaign, IL, USA, Tech. Rep. 69, 2009. [Online]. Available: http://sifaka.cs.uiuc.edu/~sondhi1/survey3.pdf

[58] M. A. Sotelo-Figueroa, H. J. P. Soberanes, J. M. Carpio, H. J. F. Huacuja, L. C. Reyes, and J. A. Soria-Alcaraz, "Improving the bin packing heuristic through grammatical evolution based on swarm intelligence," *Math. Problems Eng.*, vol. 2014, Jul. 2014, Art. no. 545191.

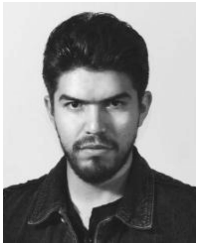
[59] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[60] L. Trujillo and G. Olague, "Automated design of image operators that detect interest points," *Evol. Comput.*, vol. 16, no. 4, pp. 483–507, 2008.

[61] M. Tuceryan and A. K. Jain, "Texture analysis," in *Handbook of Pattern Recognition and Computer Vision*, vol. 2. Singapore: World Scientific, 1993, pp. 235–276.

[62] D. H. Wolpert, "The supervised learning no-free-lunch theorems," in *Soft Computing and Industry*. Springer, 2002, pp. 25–42.

[63] X. Xu and Y. Li, "Comparison between particle swarm optimization, differential evolution and multi-parents crossover," in *Proc. Int. Conf. Comput. Intell. Secur.*, 2007, pp. 124–127.



VALENTÍN CALZADA-LEDESMA received the B.Sc. degree in computer systems engineering and the M.Sc. degree in computer science from the Instituto Tecnológico de León in 2009 and 2013, respectively. He is currently pursuing the Ph.D. degree in computer science with the Tijuana Institute of Technology, Baja California, Mexico. In 2012, he carried out a research stay at the Computing Research Center, National Polytechnic Institute, Mexico. He has authored several published papers. His research interests include computer vision, evolutionary algorithms, and image processing.

HÉCTOR J. PUGA-SOBERANES was born in Mexico, in 1960. He received the B.Sc. degree in physics and mathematics from the National Polytechnic Institute, Mexico, in 1993, and the M.Sc. and Ph.D. degrees in physics (optics) from the University of Guanajuato, Mexico, in 1995 and 2002, respectively. Since 2002, he has been with the Instituto Tecnológico de León. He has been a member of the National Researchers System, National Council of Science and Technology, Mexico, since 2004. His main research interests are optic metrology and intelligent systems, in particular, optimization with heuristics, metaheuristic, and hyper-heuristics. His current work includes SVMs and evolutionary algorithms.

MANUEL ORNELAS-RODRÍGUEZ received the B.Sc. degree in electrical-mechanical engineering from the Instituto Tecnológico de León, Mexico, in 1991, and the Ph.D. degree in physics (optics) from the University of Guanajuato, Mexico, in 2002. Since 2002, he has been a Researcher with the Instituto Tecnológico de León. His research interests include optimization, pattern recognition, and computer vision.

ALFONSO ROJAS-DOMÍNGUEZ received the Ph.D. degree in electronics engineering from the University of Liverpool, U.K., in 2007. Since 2014, he has been a Research Fellow with the National Council of Science and Technology, Mexico. He has been a member of the National Researchers System. He has authored works with over 400 citations to date. His main interests are in machine learning, computational intelligence, and artificial vision.

JUAN MARTÍN CARPIO-VALADEZ was born in León, Guanajuato, Mexico, in 1961. He received the B.Sc. degree in mathematics from the Autonomous University of Nuevo Leon, Mexico, in 1985, the M.Sc. and Ph.D. degrees in physics (optics) from the University of Guanajuato, Mexico, in 1987 and 1995, respectively. He has been a member of the National Researchers System, National Council of Science and Technology, Mexico, since 1992. Since 1994, he has been with the Instituto Tecnológico de León. His main research interests are optic metrology and data fitting with functions and orthogonal polynomials and intelligent systems, in particular, optimization with heuristics, metaheuristic and hyper-heuristics. His current work includes SVMs and evolutionary algorithms.

ANDRÉS ESPINAL received the B.Sc. degree in computer systems engineering and the M.Sc. degree in computer science from the Instituto Tecnológico de León in 2009 and 2011, respectively, and the Ph.D. degree in computer science from the Tijuana Institute of Technology in 2017. He is currently a full-time Professor with the Organizational Studies Department, University of Guanajuato, Mexico. His research interests include evolutionary algorithms, artificial neural networks, computer vision, image processing, and bio-inspired algorithms.

JORGE A. SORIA-ALCARAZ received the B.Sc. degree in computer systems engineering and the M.Sc. degree in computer science from the Instituto Tecnológico de León in 2008 and 2010, respectively, and the Ph.D. degree in computer science from the Tijuana Institute of Technology in 2015. He has carried out research stays at the Monterrey Institute of Technology and Higher Education in 2009 and at the University of Stirling, U.K., in 2014. He is currently a full-time Professor with the Organizational Studies Department, University of Guanajuato. His research interests include evolutionary algorithms, autonomous search, hyper-heuristics, adaptive operator selection, heuristic optimization, and bio-inspired algorithms.

MARCO A. SOTELO-FIGUEROA received the B.Sc. degree in computer systems engineering and the M.Sc. degree in computer science from the Instituto Tecnológico de León in 2006 and 2010, respectively, and the Ph.D. degree in computer science from the Tijuana Institute of Technology in 2015. He is currently a full-time Professor with the Organizational Studies Department, University of Guanajuato, Mexico. His research interests include evolutionary algorithms, heuristic optimization, numeric optimization, and bio-inspired algorithms.