Automatically Evolving Rotation-invariant Texture Image Descriptors by Genetic Programming

Harith Al-Sahaf, *Student Member, IEEE,* Ausama Al-Sahaf, Bing Xue, *Member, IEEE,* Mark Johnston, *Member, IEEE,* and Mengjie Zhang, *Senior Member, IEEE,*

***Abstract*—In computer vision, training a model that performs classification effectively is highly dependent on the extracted features, and the number of training instances. Conventionally, feature detection and extraction are performed by a domain-expert who, in many cases, is expensive to employ and hard to find. Therefore, image descriptors have emerged to automate these tasks. However, designing an image descriptor still requires domain-expert intervention. Moreover, the majority of machine learning algorithms require a large number of training examples to perform well. However, labelled data is not always available or easy to acquire, and dealing with a large dataset can dramatically slow down the training process. In this paper, we propose a novel Genetic Programming based method that automatically synthesises a descriptor using only two training instances per class. The proposed method combines arithmetic operators to evolve a model that takes an image and generates a feature vector. The performance of the proposed method is assessed using six datasets for texture classification with different degrees of rotation, and is compared with seven domain-expert designed descriptors. The results show that the proposed method is robust to rotation, and has significantly outperformed, or achieved a comparable performance to, the baseline methods.**

***Index Terms*—Genetic Programming, Classification, Image Descriptor, Keypoint detection, Feature extraction.**

I. Introduction

IMAGE analysis in computer vision and pattern recognition is an essential task in a wide variety of applications such as cancer and fracture detection (medical), battle field analyses (military), vegetation detection (environment), and robot navigation (robotics) [1]–[3]. Analysing the content of an image to detect an object or a region of interest remains a challenging task that has attracted many researchers over many decades. Training a model that is capable of performing the detection or classification tasks effectively is highly dependent on the features that are used during the training phase, and the number of training instances. Feature design aims at defining a set of features/keypoints that are important to locate an object in an image (object detection) or classify an image (image classification). Clearly, different applications require different features. For example, the eyes, nose, and mouth can be used to detect faces; whereas colour and texture features are used in image segmentation to differentiate between different objects or regions in an image. Feature detection aims at finding those previously designed features; this is also known as keypoint detection [4]. Locating corners and edges is a typical example of keypoint detection. Feature extraction aims at transforming raw pixel values of a detected keypoint or the region surrounding that keypoint into a reduced domain (a single value) [5]. Calculating the mean, standard deviation, thickness of a line, and measuring the distance between two pixels/positions are some examples of feature extraction. It is important to notice that different features can be extracted from the same region via applying different operators such as measuring the size and angle of a detected corner, or the homogeneity of a region. However, not all the extracted features are important or relevant. Hence, removing irrelevant and redundant features is an important task that is known as feature selection [6]. In other words, feature selection aims at selecting only a subset of the extracted features. Last but not least, in computer vision and pattern recognition, an image descriptor is a model that performs both keypoint detection and feature extraction. Hence, the raw pixel values are the input of a descriptor, whereas the output is a feature vector.

In this paper, the term *keypoint* is used to refer to a region of interest (e.g. corner, line, or spot); whilst the term *feature* is used to refer to a property extracted from a region (keypoint).

Tuytelaars and Mikolajczyk [7] have classified image keypoints into *local* and *global*. The former (local) is concerned with detecting image patterns that are distinct from their adjacent neighbourhood. Examples of local keypoints are texture, colour, shape, and intensity. Local keypoints can have a specific semantic interpretation in different applications, for example using edge detection techniques to detect roads in aerial images [7]. The robustness to image deformations in representing objects in an image is a key characteristic of local keypoints [8], and thus it has been used by researchers in many applications. Global keypoints are concerned with detecting a representative keypoint for an image in its entirety and do not discriminate between the background and foreground. The colour histogram is a typical example of a global keypoint and is very popular in image indexing and retrieval [9].

Conventionally, the process of detecting and extracting a reliable set of features is performed by a domain-expert. However, this process suffers from three potential difficulties. First, domain knowledge of the task at hand is required in order to detect important and effective keypoints. Second, domain-experts are not always available and are very expensive to employ. Third, extracting features from those detected keypoints needs to be carefully handled to ensure the robustness

H. Al-Sahaf, A. Al-Sahaf, B. Xue, and M. Zhang are with the School of Engineering and Computer Science, Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand. E-mail: {Harith.Al-Sahaf,Ausama.Al-Sahaf,Bing.Xue,Mengjie.Zhang}@ecs.vuw.ac.nz.

M. Johnston is with the Institute of Science and the Environment, University of Worcester, Worcester, WR2 6AJ, United Kingdom. E-mail: m.johnston@worc.ac.uk.

of these features to image deformations. These difficulties have attracted increasing attention and many methods have been introduced in the literature to automate keypoint detection and feature extraction in order to mitigate or overcome those difficulties [10]–[19]. Therefore, image descriptors have emerged that aim at detecting keypoints and extracting image features, which play vital roles in computer vision and pattern recognition.

Similar to other fields of machine learning, the success in performing many tasks in computer vision can be subject to, and highly dependent on, the goodness of the features. Hence, due to the ability of image descriptors to detect and extract useful information, i.e., keypoints and features, they have been widely deployed in computer vision as a prior step in object detection [20], [21], object recognition [22], image classification [23], image registration [24], and image segmentation [25]. Some typical examples of such descriptors are Haralick texture features [10], Local Binary Pattern (LBP) [11], Scale-Invariant Feature Transform (SIFT) [12], Speeded-Up Robust Features (SURF) [13], Fast Retina Keypoint (FREAK) [17], and KAZE features [18]. However, the majority of currently existing image descriptors have four major limitations. First, the process of designing image descriptors requires domain-expert intervention to select some prominent keypoints that are intended to provide useful information such as corners, edges, gradients and gradient directions, spots, and line segments. Second, altering a component of an image descriptor, e.g., the window size and number of neighbouring pixels, can be very challenging and an expert is almost always required to perform this task. Third, major changes are required in order to utilise a descriptor to handle different image deformations such as rotation and scale. Fourth, the majority of these descriptors perform multiple operations, e.g., compute the derivatives, apply Gaussian smoothing via convolution, and calculate the Histogram of Gradients (HoG) in SIFT, before a feature vector for an instance can be generated; this potentially increases the complexity and slows down the overall process.

Since the late 1990s, Genetic Programming (GP) [26] has been used to automatically evolve/extract/construct image keypoints and features [5], [27]–[35], showing good potential in this direction. The method proposed by Ebner and Zell [36] is one of the earliest works employing GP to automatically evolve an interest point detector. Similarly, Trujillo and Olague [37] have used GP to synthesise an interest point detector. They extended this work in [38] to improve the performance of the evolved points detector taking into consideration the global separability and geometric stability of the detected points. Olague and Trujillo [39] used GP to evolve image operators for detecting interest points in an image. Motivated by the success of [37], [39], Perez *et al.* [40] proposed several methods, where GP is used as a strategy to evolve image descriptors for object detection tasks. The main focus of Liu *et al.* [41] is on evolving a spatio-temporal descriptor for human action recognition by employing GP techniques. Shao *et al.* [42] proposed a multi-objective GP methodology for the task of feature learning in image classification.

Most of the aforementioned GP-based methods combine image processing operators such as convolution, derivatives and filtering, to detect a specific type of keypoint such as a corner. Having a system that is capable of detecting other types of keypoints (e.g. line ends, spots, line segments) similar to LBP, and automatically constructing models to synthesise image descriptors could be more effective at generating useful information. Moreover, it will be more efficient if the system is capable of automatically identifying keypoints without domain knowledge.

On the other hand, having too many instances in the training set imposes a heavy load on any algorithm to iterate over those instances and can dramatically slow down the training process. Ideally, all instances of one class must share a distinctive set of attribute values compared to instances from all other classes. For example, regardless of the gender of a person, almost all human faces have two eyes, one nose, and one mouth, and these features have been widely used for face detection tasks. Hence, if the system can automatically detect those features or even better ones [43] using a small number of instances, the learning process will speed-up. Motivated by this idea, in [44] GP has been used to automatically evolve an LBP-like image descriptor (GP-criptor) using the raw pixel values and only two instances per class. Moreover, promising results were achieved for texture classification as highlighted in [44]. However, the evaluation of GP-criptor is rather limited since only two datasets are used and the images to be classified are rotation free. Testing the method on images with rotations has revealed the inability of this method to evolve a rotation-invariant model and poor performance was observed. Therefore, the proposed method in this study is specifically designed to tackle the rotation problem and evolve a rotation-invariant descriptor by replacing the terminal set of GP-criptor with a set of rotation-invariant feature extraction functions.

A. Goals

The aim of this paper is to develop a new GP approach to automatically constructing rotation-invariant image descriptors that can detect good keypoints and extract informative features simultaneously for texture image classification. Instead of using a large number of instances to train/learn a classifier as in most existing supervised approaches, the proposed approach will use only two instances per class in the training set. To achieve automatic construction of rotation-invariant image descriptors, new terminal and function sets, and a fitness function need to be developed. The new image descriptors automatically constructed by GP will be examined and compared with seven state-of-the-art domain-expert designed image descriptors on ten commonly used learning/classification methods on six texture image datasets of varying difficulty with different rotations. Specifically, we will investigate the following objectives:

- Develop a new terminal set and a new function set to allow the proposed GP system to handle the rotation variation in texture images for texture classification and automatically evolve/construct image descriptors from a set of training instances;
- Develop a new fitness function that can effectively use only two labelled instances per class to evolve image descriptors and extract features for texture classification;

- Investigate whether the GP-evolved image descriptors can achieve similar or even better performance than the seven state-of-the-art domain-expert designed image descriptors;
- Investigate whether the image features evolved/generated by the proposed GP method can improve the performance of different types of learning/classification methods; and
- Investigate to what extent the evolved image descriptors/genetic programs can be interpreted by humans.

Note that the proposed method represents a substantial extension to a recent work [44] (CEC 2015 Best Overall Paper Award). Compared with [44], the proposed method in this paper can effectively deal with image rotations. Due to the increased difficulty of the task, the terminal/function sets, the fitness measures, and experiment design have been newly developed and/or substantially extended. An interpretation of the evolved programs is also provided in this paper.

B. Organisation

The remainder of the paper is organised as follows. A brief background and a survey of related work are presented in Section II. The proposed method is discussed in Section III. The experiment design is explained in Section IV. The results are presented and discussed in Section V. To provide some in-depth analysis, an example program evolved by the proposed method is extensively examined in Section VI. The conclusions of this study and some recommendations for future work are presented in Section VII.

II. Literature Survey

This section comprises two parts. A brief background on the methods directly related to this study is provided in the first part. The second part discusses GP systems for image-related work including feature detection, feature extraction, and classification.

A. Background

Here, a brief background on some of the work most directly related to this study in the literature is provided.

1) Local Binary Pattern (LBP): In computer vision and pattern recognition, *local binary pattern* (LBP) [11] is one of the most widely used descriptors for detecting and extracting image features. LBP aims at detecting image keypoints, and generates a histogram (feature vector) that corresponds to the distribution of those keypoints [45]. Conventional LBP operates by scanning the pixels of the image using a sliding window and generates a binary code based on the differences between the central pixel of the window and its circular equidistant neighbours. The distance is specified by the radius parameter (r), whereas the number of considered neighbours within the window is controlled by the pixel parameter (p)

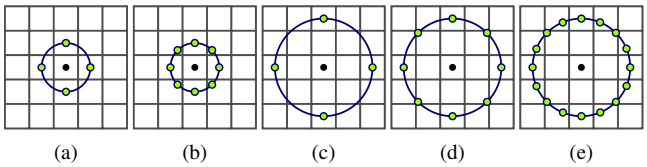


Fig. 1. Illustration of the LBP parameters (a) $\text{LBP}_{4,1}$, (b) $\text{LBP}_{8,1}$, (c) $\text{LBP}_{4,2}$, (d) $\text{LBP}_{8,2}$, and (e) $\text{LBP}_{16,2}$.

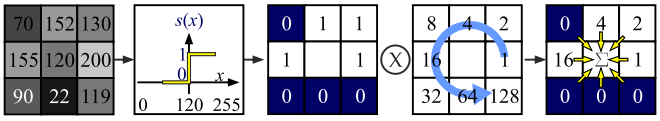


Fig. 2. Illustration of the LBP main steps.

as depicted in Fig. 1. The formal representation of the LBP operator is defined as follows:

$$\text{LBP}_{p,r} = \sum_{i=0}^{p-1} s(g_i - g_c)2^i, \quad s(x) = \begin{cases} 0, & x < 0 \\ 1, & \text{otherwise} \end{cases} \tag{1}$$

$$g_i = I(\mathbf{x}_i, \mathbf{y}_i) \tag{2}$$

$$\mathbf{x}_i = \mathbf{x}_c + r\cos(2\pi i/p) \tag{3}$$

$$\mathbf{y}_i = \mathbf{y}_c - r\sin(2\pi i/p) \tag{4}$$

where $I(\mathbf{x}_i, \mathbf{y}_i)$ is the i^{th} pixel at the $(\mathbf{x}_i, \mathbf{y}_i)$ coordinate of image I, the coordinate of the central pixel of the current window is denoted by $I(\mathbf{x}_c, \mathbf{y}_c)$, and g_c and g_i are, respectively, the value/intensity of the central and i^{th} neighbouring pixels.

The process comprises four steps at each position of the sliding window as presented in Fig. 2. First, the value of each neighbouring pixel is subtracted from that of the central pixel. Second, each negative value is substituted with a 0, otherwise a 1 is substituted. The combination of these two steps represents a thresholding operator where the value of the central pixel is used as the threshold. Third, the values (0s and 1s) are used to form a binary code. Fourth, the binary code is converted into a decimal value, and the count of the corresponding bin in the histogram is incremented by 1.

Ojala *et al.* [46], [47] have classified LBP codes into *uniform* and *non-uniform*. A code is designated as uniform ($\text{LBP}^{u2}_{p,r}$) if circularly it has no more than two bitwise transitions from 1 to 0 and vice versa as shown in Fig. 3. The following formula is used to calculate the number of bitwise transitions in a code:

$$U(\text{LBP}_{p,r}) = |s(g_{p-1} - g_c) - s(g_0 - g_c)| + \sum_{i=1}^{p-1} |s(g_i - g_c) - s(g_{i-1} - g_c)| \tag{5}$$

where a code is said to be uniform if $U(\cdot) \leq 2$. Considering only uniform codes reduces the length of the feature vector from 2^p to $p(p-1)+3$ bins. Formally, the value of the b^{th} bin in a histogram (feature vector) is calculated as:

$$H(b) = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} \delta\left(\text{LBP}^{u2}_{p,r}(i,j), b\right), \quad b \in [0, B] \tag{6}$$

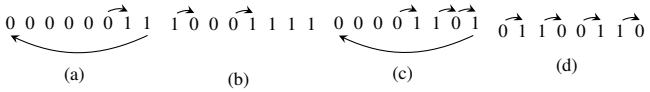


Fig. 3. Examples of (a) and (b) uniform codes, and (c) and (d) non-uniform codes. The values of the uniformity measure are, correspondingly, $U\left(00000011\right) = 2$, $U\left(10001111\right) = 2$, $U\left(00001101\right) = 4$, and $U\left(11100110\right) = 4$.

$$\delta\left(\alpha,\beta\right) = \begin{cases} 1, & \alpha = \beta \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where M and N are, respectively, the width and height of the image, B is the maximum number of bins in the histogram, and $\text{LBP}^{u2}_{p,r}\left(i,j\right)$ is the LBP code generated from positioning the sliding window at pixel coordinates (i,j).

To tackle rotations variation, a *rotation-invariant* LBP ($\text{LBP}^{ri}_{p,r}$) is also introduced by Ojala *et al.* [47]. The idea is to circularly rotate the code until the smallest value is found as presented in Equation (8).

$$\text{LBP}^{ri}_{p,r} = \min\left(ROR\left(\text{LBP}_{p,r}, x\right)\right), \quad x = 0, \ldots, p-1 \tag{8}$$

Here, the $ROR\left(\cdot,\cdot\right)$ function performs a bitwise right shift operation on the first argument (binary code) equal to the number specified by the second argument.

Then $\text{LBP}^{ri}_{p,r}$ is combined with $\text{LBP}^{u2}_{p,r}$ to generate a potentially more powerful feature vector than that generated by conventional $\text{LBP}_{p,r}$, which is indicated as $\text{LBP}^{riu2}_{p,r}$ and the formal definition is as presented in Equation (9).

$$\text{LBP}^{riu2}_{p,r} = \begin{cases} \sum_{i=0}^{p-1} s\left(g_i - g_c\right), & U\left(\text{LBP}_{p,r}\right) \leq 2 \\ p+1, & \text{otherwise} \end{cases} \tag{9}$$

As discussed above, the two parameters r (radius) and p (number of considered neighbouring pixels) can significantly affect the design of LBP. Altering the number of considered neighbouring pixels, or the formula to generate the code, requires human intervention that in many cases can be a very difficult task. Moreover, making the descriptor robust to rotation makes the task even more difficult. Hence, the method proposed in this study is designed to address these difficulties by *automatically synthesising* a set of suitable formulas, without any human intervention or background knowledge, to form a rotation-invariant image descriptor.

2) Completed Local Binary Pattern (CLBP): Typically, only the sign is considered to generate the histogram in LBP as discussed above. Guo *et al.* [48] showed that additional discriminant power is achieved by utilising the *magnitude* (CLBP_M) and the value of the central pixel (CLBP_C) along with the sign (CLBP_S), and hence, they proposed three operators:

$$\text{CLBP\_S}_{p,r} = \sum_{i=0}^{p-1} s(g_i - g_c)2^i, \quad s\left(x\right) = \begin{cases} 0, & x < 0 \\ 1, & \text{otherwise} \end{cases} \tag{10}$$

$$\text{CLBP\_M}_{p,r} = \sum_{i=0}^{p-1} s(m_i - g_c)2^i, \quad m_i = |g_i - g_c| \tag{11}$$

$$\text{CLBP\_C}_{p,r} = s(g_c - c_I) \tag{12}$$

where m_i is the magnitude of the i^{th} pixel, which represents the absolute difference between the intensity of that i^{th} pixel (g_i) and the central pixel intensity (g_c), and c_I is the average intensity, i.e., grey level, of the entire image. Clearly, $\text{CLBP\_S}_{p,r}$ is equivalent to conventional $\text{LBP}_{p,r}$.

3) Local Binary Count (LBC): Inspired by LBP, Zhao *et al.* [49] proposed the *local binary count* (LBC) descriptor. The main difference between LBP and LBC is that the code generated at each pixel (i.e. window position) is encoded into a decimal value in LBP, whereas merely the number of 1's are counted in LBC. Hence, formally LBC is defined as:

$$\text{LBC}_{p,r} = \sum_{i=0}^{p-1} s\left(g_i - g_c\right), \quad s\left(x\right) = \begin{cases} 0, & x < 0 \\ 1, & \text{otherwise} \end{cases} \tag{13}$$

where, p, r, g_i, g_c, and $s\left(\cdot\right)$ have their corresponding meanings to those symbols in $\text{LBP}_{p,r}$.

Another core difference between LBP and LBC is that the local structure information of the pattern is maintained in LBP which is not the case in LBC due to the fact that only the number of counted bits is considered while the position information is discarded.

4) Completed Local Binary Count (CLBC): A *completed local binary count* (CLBC) is proposed in [49] to mimic CLBP. Hence, the magnitude ($\text{CLBC\_M}_{p,r}$) and centre pixel ($\text{CLBC\_C}_{p,r}$) are also considered in addition to the sign ($\text{CLBC\_S}_{p,r}$). Formally, these three operators are defined as:

$$\text{CLBC\_S}_{p,r} = \sum_{i=0}^{p-1} s\left(g_i - g_c\right), \quad s\left(x\right) = \begin{cases} 0, & x < 0 \\ 1, & \text{otherwise} \end{cases} \tag{14}$$

$$\text{CLBC\_M}_{p,r} = \sum_{i=0}^{p-1} s\left(m_i - g_c\right), \quad m_i = |g_i - g_c| \tag{15}$$

$$\text{CLBC\_C}_{p,r} = s\left(g_c - c_I\right) \tag{16}$$

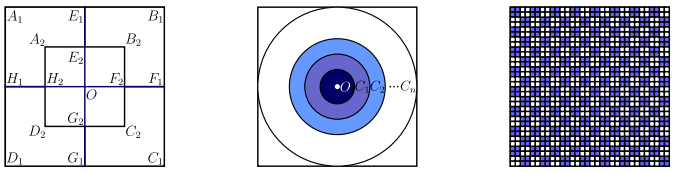
5) Haralick Texture Features: Haralick *et al.* [10] proposed a set of operators based on the *grey-level co-occurrence matrix* (GLCM) that have been widely adopted by researchers in pattern recognition and computer vision. Each matrix in GLCM has size $L \times L$, where L is the number of grey levels, and is generated by considering the occurrences of the adjacent pixels in predefined offset (τ) and angle (θ). Then a set of features are calculated from those matrices that were designed to detect the structure characterised by the keypoints. Following are some of those broadly used features.

$$\text{Contrast} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} \left(i-j\right)^2 f\left(i,j\right) \tag{17}$$

$$\text{Homogeneity} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} \frac{f\left(i,j\right)}{1+|i-j|} \tag{18}$$

$$\text{Energy} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} f\left(i,j\right)^2 \tag{19}$$

$$\text{Dissimilarity} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} |i-j|\, f\left(i,j\right) \tag{20}$$



(a) (b) (c)

Fig. 4. Three schemes to divide an image in DIF (a) rectilinear, (b) circular, and (c) pixel features [5].

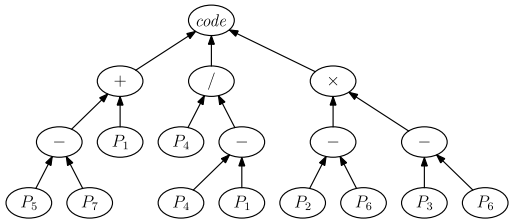


Fig. 5. An example of a program evolved by GP-criptor.

TABLE I
PIXEL STATISTICS OF THE RECTILINEAR METHOD.

Regions of interest	Features		Regions of interest	Features	
	μ	σ		μ	σ
Square $A_1B_1C_1D_1$	F_1	F_2	Square $A_2B_2C_2D_2$	F_{11}	F_{12}
Quadrant $A_1E_1OH_1$	F_3	F_4	Horizontal line H_1F_1	F_{13}	F_{14}
Quadrant $E_1B_1F_1O$	F_5	F_6	Horizontal line H_2F_2	F_{15}	F_{16}
Quadrant $H_1OG_1D_1$	F_7	F_8	Vertical line E_1G_1	F_{17}	F_{18}
Quadrant $OF_1C_1G_1$	F_9	F_{10}	Vertical line E_2G_2	F_{19}	F_{20}

$$\text{Entropy} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} f\left(i,j\right)\left[-\log_2 f\left(i,j\right)\right] \tag{21}$$

$$\text{Correlation} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} \frac{\left(i-\mu_i\right)\left(j-\mu_j\right) f\left(i,j\right)}{\sigma_i\sigma_j} \tag{22}$$

Here, the function $f\left(\cdot,\cdot\right)$ returns the value of the specified cell from the matrix, and the mean and standard deviation of the i^{th} row are, respectively, denoted as μ_i and σ_i. Similarly, μ_j and σ_j denote, correspondingly, the mean and standard deviation of the j^{th} column.

6) Domain-Independent Features: In 2003, Zhang *et al.* [5] proposed *domain-independent features* (DIF). The core idea of DIF is to extract a set of first order statistics, e.g., mean and standard deviation, from predefined image regions. Although only three schemes (rectilinear, circular, and pixel) of dividing the image are presented in their work as depicted in Fig. 4, this method is not limited and numerous schemes can be used. The features of the *rectilinear* method are shown in Table I.

7) GP Descriptor: Recently, we have proposed *Genetic Programming Image Descriptor* (GP-criptor) [44] where GP is utilised to automatically evolve an image descriptor using the raw pixel values. GP-criptor is inspired by LBP and operates a similar scheme. However it automatically evolves a set of formulas to replace the expert-designed ones. Thus, the GP tree representation of an evolved program, virtually, comprises three parts as presented in Fig. 5. The first part is represented by the leaves, i.e., terminal nodes, of the tree that are taken from the pixels of the sliding window. Hence, each leaf node represents an index of a cell in the flattened (i.e. converting the 2D window into a single vector) sliding window such that the i^{th} index is indicated by P_i. The second part consists of a set of sub-trees each of which contains arithmetic operators ($+$, $-$, $\times$, and $/$). The third part resides at the top (the root) of the program and consists of a single node called code, which converts the output of its children into a binary code.

The work in [44] is limited since GP-criptor has only been tested using two datasets, and compared to LBP, GLCM, and DIF features. Although the results achieved are promising, GP-criptor was not designed to handle image rotations and further tests have revealed its inability to evolve rotation-invariant descriptors.

B. Related Work

GP has attracted many researchers over the last few decades to tackle image-related problems, e.g., keypoint detection, feature extraction, feature selection, and classification, in a wide variety of applications.

Song *et al.* [50], [51] proposed two GP representations for texture classification using the raw pixel values. Their methods perform multi-class classification by adopting the Static Range Selection (SRS) [52], [53] and Dynamic Range Selection (DRS) [54] approaches. In [50], the authors evaluated their methods using four classes from the commonly used *Brodatz Textures* [55] dataset and promising results have been observed.

Smart and Zhang [56] proposed a GP based approach to multi-class object classification tasks relying on the Centered Dynamic Range Selection (CDRS) and Slotted Dynamic Range Selection (SDRS) strategies. Comparing with static range selection methods by using five datasets for image classification of increasing difficulty, these two dynamic methods have outperformed the competitors especially on the more difficult problems. However, these methods were designed to evolve a classifier rather than a descriptor. Moreover, these methods require a large number of training instances to achieve a suitable level of performance.

To improve the GP search for object detection, Zhang *et al.* [57] proposed a two-phase approach that uses a sample (subset) of the training instances and a simplified fitness function in the first phase to evolve initial solutions; whereas those initial solutions, the entire set of training instances, and a complete fitness function are used in the second phase. The program size is added to the fitness function with the aim to have smaller and easier to interpret programs. Compared to conventional Artificial Neural Networks (ANN) on three datasets of increasing difficulty, their results show that the innovations introduced have improved the performance of GP in terms of effectiveness and efficiency, and the GP approach has outperformed the ANN approach. Similar to [56], the focus of [57] is on evolving a classifier rather than a descriptor and also in this case a large number of instances are required. Moreover, their method is a multi-phase approach where the output of the first phase is needed in order to start the second phase, which increases the complexity of the overall system.

A modified GP representation for multi-class object classification is proposed by Zhang and Johnston [58]. The main

idea is to evolve a program that produces multiple values, i.e., one for each class label, instead of the single value of the conventional GP individual. This approach was assessed using four multi-class object classification datasets, and the results showed it has outperformed the standard GP approach. This method tackles multi-class classification and does not operate directly on the raw pixel values; human intervention is required to perform keypoint detection and feature extraction.

Atkins *et al.* [59] proposed a three tier GP (3T-GP) program representation for image classification. The lower tier performs image filtering, with feature extraction in the middle tier, and classification in the upper tier. The 3T-GP achieved a comparable performance to that of the domain-specific hand-crafted features.

Motivated by the promising results of the 3T-GP method, Al-Sahaf *et al.* [43] have further studied this multi-tier approach and concluded that better performance can be achieved by removing the filtering tier. Hence, their two-tier GP (2T-GP) approach is quite similar to the 3T-GP approach but with more terminals and functions added [30], [43]. The 2T-GP has been tested using datasets for different applications and it was shown to outperform the competitor methods [43]. Moreover, using the features extracted by 2T-GP improved the performance of the different classifiers compared to the use of domain-specific hand-crafted features [30].

Both the 3T-GP and 2T-GP methods are closely related to the method proposed in this study, as they automatically detect prominent regions (keypoints) and extract different features from those detected regions. Moreover, in [30] it was shown that feeding the features extracted by 2T-GP to a variety of classifiers significantly improved the performance of those classifiers compared to domain-specific features. However, both 3T-GP and 2T-GP require a large number of training instances [60], and extending those methods to handle multi-class classification tasks requires substantial changes.

Fu *et al.* [61], [62] used GP for edge detection. The edge represents an important feature which many algorithms in computer vision rely on, when performing detection and segmentation. Their results show that cleaner and more edges of interest were detected compared to well-known algorithms such as Sobel and Canny [63]. Those methods (i.e. [61], [62]) are closely related to the proposed method in this paper; however, our new method is not limited to detecting only a specific type of feature.

Albukhanajer *et al.* [31] proposed a multi-objective approach to extracting image features that are robust to noise and invariant to geometric deformations, e.g., illumination, rotation, and scale, by optimizing the functionals in the trace transform[1]. In this method, the system automatically combines different trace, diametric, and "circus functionals" in order to minimise the within-class variance and maximise the between-class variance. Their experiments on two datasets showed that the method is robust to noise and geometric deformations.

In summary, most of the existing methods in the literature involve human intervention, i.e., they need human experts to design the keypoints and manually write programs to extract features from those keypoints. They typically perform keypoint detection and feature extraction separately, and if some important keypoints/regions are missed in the first stage, it would be almost impossible to extract good features from the missed keypoints or regions. In most image classification systems, a large number of training examples/instances are required to build or train a good classifier. Many of the existing methods or image descriptors cannot effectively classify images with different rotations. This paper will develop a new approach using GP to tackle these issues. The next section will describe the new method.

[1]Trace transform is a general representation of the Radon transform, which uses straight lines to calculate the image functionals [64].

III. The Proposed Method

This section provides a detailed discussion on the proposed *rotation-invariant* GP-criptor (GP-criptor^{ri}) method. The section starts by presenting an overview of the algorithm to evolve a program in order to highlight the key components of GP-criptor^{ri}, and how the evolved program is evaluated. Then the program structure, i.e., terminal and function sets, fitness measure, and feature vector extraction process are discussed.

The proposed method operates directly on the raw pixel values, and therefore it does not require human intervention to provide a set of predefined/extracted features. Unlike methods designed by domain experts, the proposed method does not use domain knowledge to detect a specific set of keypoints such as lines, corners, spots, or homogeneous regions. Instead, GP-criptor^{ri} automatically discovers good keypoints that vary in their frequency of appearance between the instances of the different classes. Moreover, GP-criptor^{ri} does not require human intervention to manually combine the detected keypoints (like in LBP and GLCM) and design them to be rotation-invariant. Instead, this method automatically synthesises a set of mathematical formulas to accomplish this task. Another key feature of the proposed method is that it does not need a large number of training instances to evolve a descriptor, which makes it suitable for applications where the labelled data is limited. In fact, being capable of operating on just a few instances has a large impact on reducing the training costs, i.e., memory and CPU time.

A. The Overall Algorithm

Fig. 6 presents an overview of the overall algorithm. The system divides the total number of instances of each class equally between the training and test sets. Note that only two instances per class are randomly selected from the training set and fed to the GP system to evolve the programs (i.e. descriptors), whereas the rest of the training samples are discarded. From the diagram presented in Fig. 6, we can see that, at the end of the evolutionary process, the system returns the best evolved program which represents an automatically evolved image descriptor. The instances that were used during the evolutionary process (i.e. the two randomly selected instances per class) are fed to the evolved descriptor to generate a special set called the *knowledge-base* ($\mathbf{D}$). This set is the transformed training subset and will be used to train a classifier. In other words, the two instances per class randomly selected to form

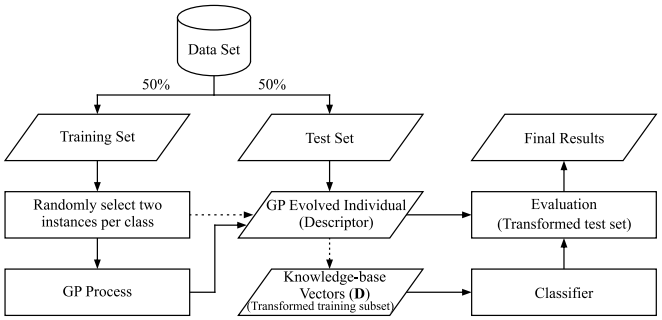


Fig. 6. An overview of the overall algorithm.

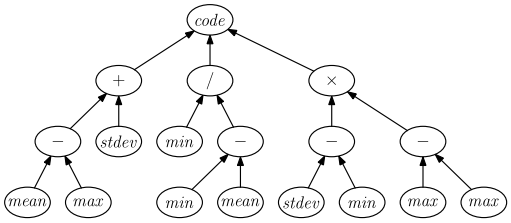


Fig. 7. The program representation of an individual evolved by GP-criptor^{ri}.

the training set are used to evolve an individual and later to train a classifier. The evolved descriptor is then used to generate the feature vectors (transformed dataset) for the unseen (test) data. These feature vectors are fed to a trained classifier (using $\mathbf{D}$ as training set) in order to assess the performance of the evolved descriptor on the unseen data.

B. Program Representation

In GP, the individual programs are constructed from elements of the terminal and function sets. In this study, tree based GP [26] is used to represent an individual (i.e. image descriptor) evolved by GP-criptor^{ri}, where the terminal nodes, i.e., leaves, are taken from the terminal set, and all non-terminal nodes are drawn from the function set. Fig. 7 depicts an example of a GP-criptor^{ri} evolved individual. Moreover, *strongly-typed GP* (STGP) [65] is used to introduce restrictions on the nodes. Each individual is a set of synthesised formulas that are used to extract the feature vector (more details in Section III-D).

The terminal set consists of four nodes: $min\,(\vec{x})$, $max\,(\vec{x})$, $mean\,(\vec{x})$, and $stdev\,(\vec{x})$, which are functions that respectively return the minimum, maximum, mean, and standard deviation values of the elements of a vector. The intuition behind choosing these functions is their order-independent property when extracting features. In other words, shuffling the values of the vector will not affect the results returned by those functions. This is very important to handle the rotation variants of the pixels. The terminal nodes take a vector of integer values, and return a single floating point value.

The terminal set is a key difference between GP-criptor and GP-criptor^{ri}. In GP-criptor, the leaf nodes of an evolved program are the original pixel values in a randomly selected index of the sliding window as presented in Fig. 5; whereas the leaf nodes in GP-criptor^{ri} are calculated statistics of the pixel values of the sliding window as shown in Fig. 7.

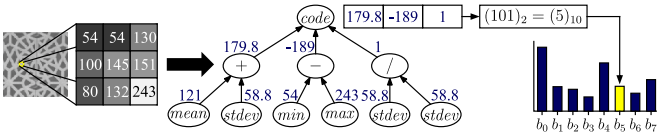


Fig. 8. An example demonstrates the process of converting an instance to a feature vector.

The function set is made up of five nodes: $code$ and the four arithmetic operators $+$, $-$, $/$, and $\times$. The arithmetic operators have their regular meaning, however $/$ is protected so that it returns zero if the denominator is zero in order to avoid the "division by zero" problem. Apart from $code$, those functions take two input arguments and return a single output. Moreover, the input and output types are floating-point values, and hence, the output of one node can be an input of another node. More operators can be used such as trigonometric functions and logical operators, which represents another key benefit of the GP-criptor^{ri}: it has more flexibility than those descriptors designed by domain-experts. The $code$ node $code$, on the other hand, takes a predefined number (specified by the user) of arguments and returns a binary code. This node cannot appear as a child of any other node due to the type mismatch between its output and the output of other function nodes. The $code$ node resides at the root of the individual tree, and each individual has only one $code$ node. This node converts its inputs into binary by using 0 as a threshold as demonstrated in Fig. 8. The generated codes are used to construct the feature vector (more details in Section III-D).

C. Fitness Measure

Typically, the classification accuracy is used as the fitness measure to gauge the performance of the individual to discriminate between instances of different classes. Accuracy is defined as the ratio between the number of correctly classified instances and the total number of instances. The use of accuracy to measure the fitness is inappropriate when there are only a few examples in the training set as the algorithm will simply memorise those examples, which can increase the possibility of "over-fitting" occurring and affect the generalisability of the evolved program on the unseen data. Therefore, a different measure is needed to cope with the problem of having only a limited number of training instances. Clearly, we need a fitness function that can detect as many representative keypoints as possible that reliably separate the instances of different classes farther apart, and keeps the distances between instances of a particular class as close to each other as possible [66]. Hence, the fitness measure that is used in this study is defined as:

$$fitness = 1 - \left(\frac{1}{1 + e^{-5(D_\mathrm{b} - D_\mathrm{w})}} \right) \tag{23}$$

where D_b is the average distance of *between-class* instances calculated using Equation (24), and D_w is the average distance

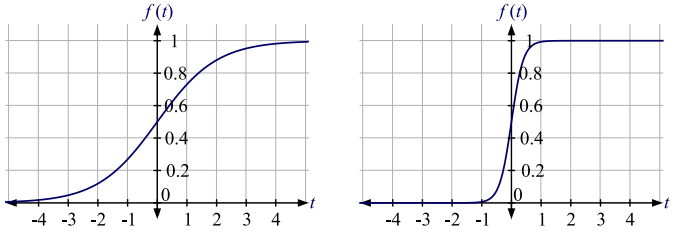


Fig. 9. Logistic function (a) $f(t) = \frac{1}{1+e^{-t}}$, and (b) $f(t) = \frac{1}{1+e^{-5t}}$.

of *within-class* instances calculated using Equation (25).

$$D_{\mathrm{b}} = \frac{1}{z\,(z-n)} \sum_{\substack{\mathbf{u}^{\alpha},\mathbf{v}^{\beta}\in\mathbf{S}_{\mathrm{tr}} \\ \forall\vec{u}\in\mathbf{u}^{\alpha} \\ \forall\vec{v}\in\mathbf{v}^{\beta}}} \chi^2\left(\vec{u},\vec{v}\right), \quad \alpha,\beta\in[1,c]\,, \alpha\neq\beta \tag{24}$$

$$D_{\mathrm{w}} = \frac{1}{z\,(n-1)} \sum_{\substack{\mathbf{u}^{\alpha},\mathbf{v}^{\alpha}\in\mathbf{S}_{\mathrm{tr}} \\ \forall\vec{u}\in\mathbf{u}^{\alpha} \\ \forall\vec{v}\in\mathbf{v}^{\alpha}}} \chi^2\left(\vec{u},\vec{v}\right), \qquad \alpha\in[1,c] \tag{25}$$

Here, $\mathbf{S}_{\mathrm{tr}} = \{(\vec{x}_i, y_i)\}$ is the training set, where $\vec{x}_i \in \mathbb{R}_{\geq 0}$ is the feature vector, y_i is the class label, and $i \in \{1,\ldots,z\}$; c and n are, respectively, the total number of classes and the number of instances per class; z is the total number of instances in the training set (i.e. $c \times n$), and $\mathbf{x}^{\alpha}$ is the set of all instances of the α^{th} class in $\mathbf{S}_{\mathrm{tr}}$. The widely used $\chi^2(\cdot,\cdot)$ function measures the distance between two normalised vectors of the same length as:

$$\chi^2\left(\vec{u},\vec{v}\right) = \frac{1}{2}\sum_{i}\frac{\left(u_i - v_i\right)^2}{u_i + v_i} \tag{26}$$

where u_i and v_i are the i^{th} element in the $\vec{u}$ and $\vec{v}$ vectors, respectively. When the denominator of Equation (26) for an index i, i.e., $u_i + v_i$, is zero, the function returns zero in order to prevent the *division by zero* issue [67].

This fitness measure (Equation (23)) returns 1 and 0, respectively, in the worst and best case scenarios. Moreover, the second part of Equation (23) is a modified version of the conventional *sigmoid* function as presented in Fig. 9(a). The value 5 is included in the exponent to scale down the input range from approximately $[-5,+5]$ to approximately $[-1,+1]$ as depicted in Fig. 9(b). The motivation behind making the effective input range narrow is mainly because the $\chi^2(\cdot,\cdot)$ function returns values in the interval $[0,1]$. Therefore, using the conventional sigmoid function the results of $f(t)$ will be approximately in the interval $[0.27, 0.73]$. Therefore, including 5 is to scale the output interval to be approximately $[0,1]$.

D. Feature Vector Extraction

A core task of an individual evolved by GP-criptor^{ri} is to automatically detect keypoints and extract a feature vector from an instance, i.e., an image, being evaluated using a sliding window of predetermined size. The length of the feature vector depends on the number of children of the *code* node (the root of the individual tree). If there are q nodes in the children list of *code*, then the resulting vector for each instance is of length 2^q. As demonstrated in Fig. 8, the instance undergoes five steps at each position of the sliding window traversing the instance pixel-by-pixel row-wise starting from the top-left corner and ending at the bottom-right corner.

Step 1: The minimum, maximum, mean, and standard deviation values of the current window pixels are calculated.

Step 2: Those calculated values are fed to the terminal nodes of the individual.

Step 3: The internal (non-terminal) nodes, apart from the root node, are evaluated starting from those near the leaves by applying the corresponding operator to the list of arguments, i.e., children.

Step 4: The root node (i.e. *code*) returns a binary code by converting each of its arguments to 0 if it is negative and 1 otherwise.

Step 5: The generated binary code is converted to decimal, and the corresponding bin of the feature vector (histogram) is incremented by 1.

Clearly the *code* node in this context mimics the thresholding step of the conventional LBP descriptor. However, the latter uses the central pixel's value as a threshold; whilst 0 is used as a threshold value in the former.

IV. Experiment Design

The aim and design of the experiments are discussed in this section. The discussion also includes the datasets, methods for comparison, and parameter settings.

A. Data Sets

The proposed method is evaluated using six image datasets for texture classification. The instances of all those image datasets are grey-scale images, i.e., each pixel carries only brightness/intensity information that can be white at the strongest intensity or black at the weakest intensity [68]. Therefore, the pixel values are ranging between 0 (black) and 255 (white).

The first and second datasets in our experiments are drawn from the widely used *Brodatz Texture*[2] [55] dataset. Originally, this dataset consisted of 112 classes, each of which comprises a single 640×640 pixels grey-scale image. We have randomly selected 20 classes out of the 112, and re-sampled the original image into 84 non-overlapping tiles, i.e., sub-images, of size 64×64 pixels. Therefore, the first dataset in our experiments *Brodatz without rotation* (BrNoRo) consists of $1,680$ instances in total. To test the ability of the proposed method to handle rotation, we have rotated those original images around the centre through 360° and re-sampled every 30° giving $1,008$ instances in total (12 angles $\times$ 84 tiles) in each class that are used to form the second dataset *Brodatz with rotation* (BrWiRo) in this study.

Similarly, the third and fourth datasets are taken from the popular *Kylberg Texture*[3] [69] dataset. The Kylberg dataset

[2]Available at: http://multibandtexture.recherche.usherbrooke.ca/original_brodatz.html

[3]Available at: http://www.cb.uu.se/~gustaf/texture/

TABLE II
A SUMMARY OF THE DATASETS.

Data set	N_{classes}	$N_{\text{instances}}$	N_{rotation}	Dimensions w	Dimensions h
BrNoRo	20	1,680	0	64	64
BrWiRo	20	20,160	12	64	64
KyNoRo	28	4,480	0	115	15
KyWiRo	28	53,760	12	115	115
OutexTC00	24	480	0	128	128
OutexTC10	24	4,320	9	128	128

comprises 28 classes of different material textures, and its instances come in two flavours: *without-* and *with-rotation*. The former category forms the third dataset in this study (KyNoRo), whereas the latter category is used to form the fourth dataset (KyWiRo). Each instance in the Kylberg dataset is of size 576×576 pixels. Handling large images can easily consume the computer's physical memory and can be very time consuming to operate on; hence, in our experiments those instances have been sampled to 115×115 pixels each. There are 160 instances in each class of the without-rotation category, whereas each class of the with-rotation category consists of $1,920$ instances. The instances of the with-rotation category are the same as those of the without-rotation rotated around the centre in 12 angles ($160 \times 12 = 1,920$) between 0° and 330° with a step of size 30°.

The *Outex Texture Classification*[4] dataset [70] comprises 16 sets each of which consists of a different number and type of texture. The content of Outex_TC_00000 is used to form the fifth dataset (OutexTC00) in this study. Meanwhile, the sixth dataset (OutexTC10) is formed using Outex_TC_00010. Each of these sets is made up of 24 classes. OutexTC00 has 20 instances in each class and is fixed in terms of rotation, whereas OutexTC10 has 180 instances in each class which fall into 9 subsets of different angles between 0° and 90° (0°, 5°, 10°, 15°, 30°, 45°, 60°, 75°, and 90°). The instances of both sets are grey-scale and with size 128×128 pixels.

Table II summarises the number of classes (N_{classes}), instances ($N_{\text{instances}}$) and rotations (N_{rotation}) in each dataset, and the dimensions (w = width, and h = height) of each instance. All instances of these datasets are grey-level with 256 intensity values. Some samples of those datasets are presented in Figs. 10–12.

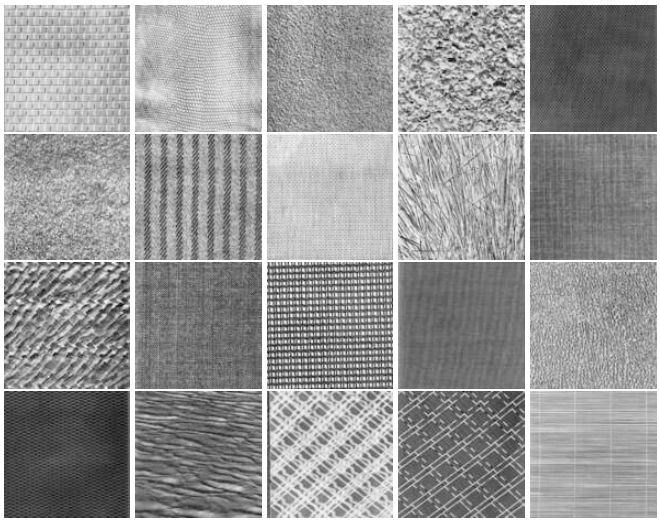


Fig. 10. Samples of the Brodatz dataset.

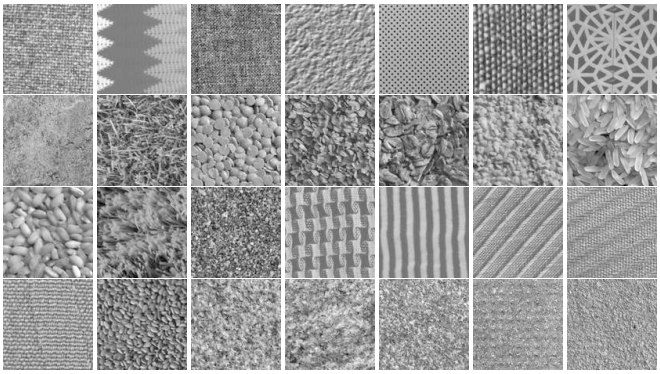


Fig. 11. Samples of the Kylberg dataset.

B. Methods for Comparison

In order to investigate the effectiveness of the proposed method, its performance is compared to the performance of the common state-of-the-art descriptors DIF, GLCM, $\text{LBP}^{u2}_{p,r}$, $\text{LBP}^{riu2}_{p,r}$, $\text{CLBP}_{p,r}$, $\text{LBC}_{p,r}$, and $\text{CLBC}_{p,r}$. Image descriptors are used as pre-processing methods to convert an instance from raw pixel values into a feature vector after detecting some keypoints. Hence, researchers broadly rely on classification or recognition to assess the goodness of a descriptor [11]–[13], [71]. In this study, we also used the classification performance to assess whether the proposed method is capable of evolving good descriptors. We have intentionally selected classifiers of different types in this study to ensure that the new method is not biased towards a specific classifier or type of classifier. Those classifiers are Support Vector Machines (SVM), Naïve Bayes (NB), Adaptive Boosting (AdaBoost), Decision Trees (J48), Random Forest (RF), Naïve Bayes/Decision Tree (NBTree), KStar (K^*), Non-Nested generalised (NNge), and Multilayer Perceptron (MLP). The implementations of these classifiers are taken from the commonly used Waikato Environment for Knowledge Analysis (WEKA) [72] software. For more details regarding these methods, see [73].

[4]Available at: http://www.outex.oulu.fi/index.php?page=classification

C. Experiments

The aim of the proposed method is to automatically evolve an image descriptor that generates distinctive feature vectors for instances belonging to different classes. Therefore, two experiments are designed each of which aims at investigating a specific aspect. On each dataset, the proposed method is executed 30 times using different random seeds, and the performance of the best evolved program at each run is recorded. Then the average performance (mean $\pm$ standard deviation) of those 30 best programs is calculated. The training instances (2 instances per class) are *randomly* selected, hence using different instances could give different results. Therefore, the process of 30 runs is repeated 10 times using

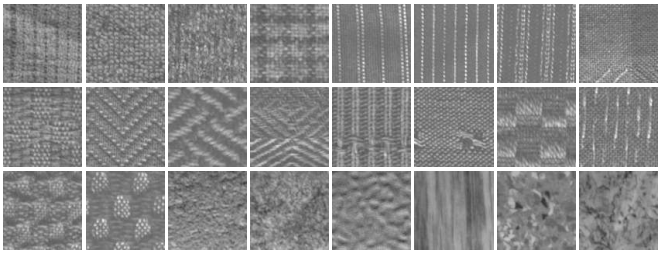


Fig. 12. Samples of the Outex TC dataset.

different instances for training each time. The experiments have been executed on the grid-computing facility provided by Victoria University of Wellington. This grid runs under the *Sun Grid Engine* (SGE) control, and consists of a large number of machines that are running Linux version 3.7.5-1-ARCH operating system with an Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz and a 8GByte of memory each. It is important to notice that the measured time in all our experiments was the CPU time and not the wall-clock time. The first experiment can be considered as a *parameter-tuning* phase, where in total we have 9 combinations of different window sizes and code lengths (details are below). Hence, we have $16,200$ runs in total, i.e., 6 (datasets) $\times$ 9 (combinations) $\times$ 10 (repetitions) $\times$ 30 (runs). Using one more window size or code length will require 900 runs to be added, which is a very time consuming process.

1) Window size and code length: In the first experiment, the impacts of changing the window size and the code length on the performance of GP-criptor^{ri} are studied. Changing the sliding window size allows a different number of pixels to contribute towards calculating the code at each position. Therefore, three window sizes are tested: 3×3, 5×5, and 7×7.

The number of bits of the binary code, on the other hand, is the only factor that specifies the feature vector length (2^q where q is the number of bits), therefore, the length is doubled for each extra bit added to the code. We have experimented with three code lengths: 7, 8, and 9.

2) Image classification: The second experiment is designed to test whether using a simple instance-based classifier, i.e., the k-Nearest Neighbour (kNN with k set to 1), with only a few learning examples can achieve comparable or even better performance than using LBP^{u2}, LBP^{riu2}, CLBP, LBC, CLPC, DIF, and GLCM with more powerful classifiers such as SVM, K*, RF, AdaBoost, NNge, NB, NBTree, J48, and MLP. The proposed method heavily relies on the between-class and within-class distances as the main criteria to evolve a good descriptor. Hence, the impact of the features generated by the proposed method on the performance of each of the aforementioned nine classifiers is also studied. This will help in identifying whether those features are biased toward a specific type (i.e. instance-based or kNN-like) of classifiers or not.

D. Parameter Settings

The methods used for comparison in this study as well as the proposed method contain a number of parameters that need to be set. The parameter settings of the proposed method are discussed first, followed by a discussion on setting those of the other methods.

TABLE III
THE GP PARAMETERS

Parameter	Value	Parameter	Value
Generations	50	Crossover Rate	0.80
Population Size	200	Mutation Rate	0.20
Minimum Depth	2	Maximum Depth	10
Selection Type	Tournament	Reproduction	Keep the best
Tournament size	5	Initial Population	Half-and-half

1) Evolutionary parameters of the proposed method: The GP evolutionary parameters are summarised in Table III. Dealing with images, in general, is an expensive task in terms of the required physical memory and processing time. This will become more problematic when multiple images need to be processed again and again (as many individuals are in each population), and over multiple generations. Hence, the number of individuals per population is relatively small (200) in order to reduce the costs, while still having a diverse population. The evolutionary process is terminated if an *ideal* individual (fitness value equal 0) is found, or the maximum number of 50 generations is reached. The probability of performing crossover is set to 80%, whereas the mutation operation is performed 20% of the time to maintain the population diversity. The best evolved program will be kept to prevent the performance of the subsequent generation from degrading. The minimum and maximum depth of an evolved program is restricted to be, respectively, 2 and 10. The commonly used *ramped half-and-half* method is used to generate the initial population. Selecting individuals for the mating process is handled using tournament selection with a tournament of size 5. These are commonly used settings in the literature [59].

2) Parameters of the baseline methods: Based on the observations of [49] and [74], the radius (r) and the number of neighbouring pixels (p) for LBP, CLBP, LBC and CLBC have been, respectively, set to 3 and 24. This combination has shown very good performance in most cases in both studies. Hence, we have also used the same settings in our experiments. For the LBP$^{u2}_{p,r}$ method, on the other hand, our experiments show that this method has achieved good performance when r is set to 1 and p to 8.

3) Parameters of the classifiers: In WEKA, most of the implemented algorithms have a predefined set of default parameters that in most cases are not the optimal ones to use. Hence, some parameter tuning and optimisation, based on either the literature or via experiments, have been applied to most of the other methods, i.e., classifiers, in this study.

In the instance-based methods, i.e., K* and NNge, the number of neighbours is set to 1 (the closest neighbour) as there are only two instances per class in the training set in our experiments. Meanwhile, the guidelines of Trenn [75] are considered to specify the network structure of MLP.

Keerthi and Lin [76] have studied the impact of using linear and non-linear kernels on the performance of SVM, and concluded that a non-linear kernel has the potential to improve SVM performance in many cases over a linear kernel. Hence,

we have used a *Radial Basis Function* (RBF) kernel in our experiments.

AdaBoost works in conjunction with another classifier [77]. Using *DecisionStump*, the default classifier in WEKA for AdaBoost, did not achieve good results; therefore, it has been replaced by a *muLti-class Alternating Decision Trees* (LADTree) classifier [78] in our experiments.

V. Results and Discussions

The results of the experiments are presented and discussed in this section.

A. Window Size and Code Length

Due to the effect of the correlation between the size of the sliding window and the code length on the performance, the results of these two factors are combined into a single set. The results of each dataset are presented in a single 3D bar chart, where the x-axis is the code length (number of bits), y-axis is the window size, and z-axis is the accuracy (%) as shown in Fig. 13.

The proposed method has achieved on average the minimum of 88.35% and 89.88% accuracy on the BrNoRo and BrWiRo datasets, respectively, using the combination of window size 7×7 and code length 7-bits as depicted in Fig. 13(a),(b). The maximum average accuracies on these datasets were achieved when the window size is set to 5×5 and a 9-bits code length, which are 90.92% and 92.49% respectively.

Similarly, using a window of size 7×7 pixels and a code of length 7-bits, GP-criptor^{ri} on the KyNoRo and KyWiRo datasets has achieved the lowest average performance that are, respectively, 85.05% and 85.96% as shown in Fig. 13(c),(d). The best average performances are scored on these two datasets when the code length is increased to 9-bits and the window size is reduced to 5×5 pixels, which are 86.66% (KyNoRo) and 88.51% (KyWiRo).

While the results of the OutexTC00 dataset show a similar pattern to that of the previous four where a minimum average performance (86.19%) is achieved with a window of size 7×7 pixels and code length 7-bits, the OutexTC10 dataset results show a different pattern as the minimum performance was achieved with a window of size 3×3 and 7-bits code length as, respectively, presented in Fig. 13(e),(f). However, the combination of a 5×5 pixels window and 9-bits code length still gives the best average accuracies 87.68% and 86.82%, respectively.

In summary, the six datasets show a similar pattern, that is, a better performance has been achieved with a code of length 9-bits than that of length 7- and 8-bits as depicted in Fig. 13(a)–(f). Similarly, the proposed method has achieved slightly better performance when the window size is 5×5 pixels than the other two experimented sizes, i.e., 3×3 and 7×7 pixels. Hence, in our subsequent experiments we have used the combination of window size 5×5 and code length 9-bits as it has been shown to give the best average performance.

B. Image Classification

The results for the second experiment on the six texture datasets are presented in Table IV. Vertically, this table comprises blocks that each correspond to one dataset, whilst horizontally this table is made up of 11 columns (one to list the descriptors and 10 for the different classifiers). The values in this table are the average accuracy percentage $\pm$ the standard deviation resulting from using a classifier (column) with an image descriptor (row).

Selecting the right statistical test is very important in order to correctly test the significance of the obtained results. As the normality test showed that the results are mostly skewed, the use of a non-parametric test is more appropriate than applying a parametric test [79]. Hence, these results are statistically tested using the *Wilcoxon signed-rank* test [80], [81] with a significance level of 5%. The statistical test has been applied twice, first, to check whether the proposed method with a simple classifier (1NN) can compete with the baseline methods (i.e. descriptors) with more powerful classifiers; and second, to test whether the new method can compete with the baseline methods using the same classifier. The symbols "$*$" and "$-$" are used to, respectively, represent significantly better and significantly worse in the first test, whereas significantly better and significantly worse in the second test are indicated by "$\uparrow$" and "$\downarrow$", respectively. For each dataset, the corresponding method with best performance for each classifier is made bold; whilst the best performance amongst all methods and classifiers is underlined.

The code length is set to 9-bits and a window of size 5×5 pixels is used as the majority of these datasets have performed well with this combination in preliminary testing.

1) BrNoRo: On the BrNoRo dataset, the proposed method with 1NN has achieved on average 90.9% accuracy which is significantly better than all other methods as presented in the first block of Table IV. Meanwhile, those features extracted by the proposed method have positive influence on the performance of all other classifiers, apart from MLP with CLBP$_{24,3}$, compared to the performances achieved using the baseline methods.

2) BrWiRo: The second block of Table IV shows the results of the BrWiRo dataset which is the rotated version of BrNoRo. The proposed method with 1NN has achieved the best performance over all other methods with more sophisticated classifiers. On average, the new method has scored 92.5% which is comparable to the performance on the without-rotation version, i.e., BrNoRo, of this dataset. This reveals the ability of the new method to handle rotations. Although the new method has slightly degraded the performance of MLP and SVM with CLBP$_{24,3}$, using the extracted features by GP-criptor^{ri} with the other classifiers shows a significant improvement in their performances compared to other image descriptors.

3) KyNoRo: The results presented in the third block of Table IV correspond to the KyNoRo dataset. The proposed method shows on average 86.7% accuracy on the unseen data of KyNoRo. Meanwhile, both CLBP$_{24,3}$ and CLBC$_{24,3}$ have significantly outperformed the new method by scoring on average 90.3% and 91.1%, respectively. The same also

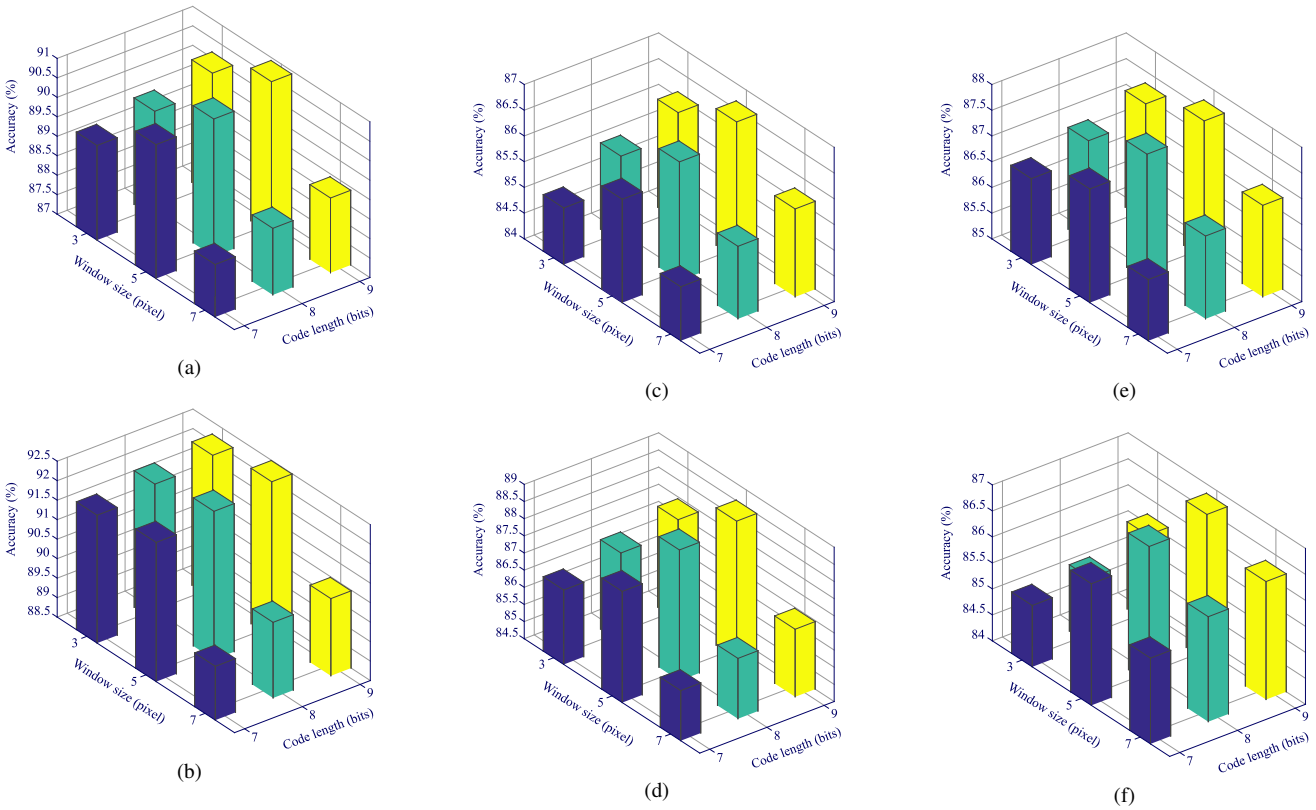


Fig. 13. The results of the first experiment, which presents the impact of the window size and code length on the performance on the (a) BrNoRo, (b) BrWiRo, (c) KyNoRo, (d) KyWiRo, (e) OutexTC00, and (f) OutexTC10 datasets.

happened when the features extracted by GP-criptor^{ri} are used with MLP, NNge, and SVM. In all other cases, the impact of the GP-criptor^{ri} features either significantly improved the performance or achieved a comparable level, i.e., best or in the top-three ranked performances, to that of the expert designed methods.

4) KyWiRo: The results of the experiments on the rotated version of KyNoRo (KyWiRo) are listed in the fourth block of Table IV. The results on this dataset are quite similar to those that have been observed on KyNoRo. The performance achieved by any classifier using the proposed method's features is either the first or in the top-three best performances compared to the use of hand-crafted descriptors.

5) OutexTC00: The results on OutexTC00 are presented in the fifth block of Table IV. The proposed method has scored the second overall best performance (the first is $\text{LBP}^{u2}_{8,1}$ with NNge) with 87.7% accuracy on this dataset, and has outperformed all the competitor methods.

6) OutexTC10: On the OutexTC10 dataset, the proposed method has achieved the best performance with 86.8% accuracy on average over all the baseline methods as shown in the sixth block of Table IV. Furthermore, 7 out of 10 classifiers, i.e., 1NN, AdaBoost, J48, K*, NB, NBTree, and RF, are ranked number one when GP-criptor^{ri} features are used. The differences are also significant in most of these cases.

C. Summary

From the results above, the following observations can be deduced.

- The proposed method has the ability to automatically evolve an image descriptor using the raw pixel values and without human intervention;
- The system uses only two instances of each class and yet it has been shown to outperform most domain-expert designed descriptors;
- The evolved descriptors do not solely detect a specific and predetermined set of keypoints, e.g., lines, corners, and spots; rather, it automatically detects a set of informative keypoints;
- The features of the new method have, in the majority of the cases, positive influence on the performance of classifiers of different types;
- Unlike domain-expert designed descriptors, the proposed method evolves a rotation-invariant descriptor that does not require human intervention to handle this issue;
- It is easy to change the parameters, e.g., the window size and the number of bits in the code, in the proposed method since they are handled automatically without human intervention, while domain-expert involvement is required to alter these parameters in other methods; and
- The program structure of the proposed method is flexible and allows different types of functions to be used for feature extraction.

TABLE IV
ACCURACY (%) OF 10 CLASSIFIERS USING 8 IMAGE DESCRIPTORS ON THE 6 TEXTURE IMAGE DATASETS (MEAN ± STANDARD DEVIATION).

	1NN	AdaBoost	J48	K*	MLP	NB	NBTree	NNge	RF	SVM
	BrNoRo									
DIF	36.8 ± 2.7∗	18.0 ± 3.6∗↑	28.5 ± 4.9∗↑	36.4 ± 3.2∗↑	35.3 ± 2.5∗↑	21.0 ± 6.4∗↑	33.4 ± 4.7∗↑	34.3 ± 3.5∗↑	31.7 ± 3.5∗↑	33.5 ± 2.8∗↑
$LBP^{u2}_{8,1}$	68.9 ± 3.8∗	37.2 ± 5.9∗↑	25.4 ± 3.9∗↑	61.9 ± 5.3∗↑	69.7 ± 2.5∗↑	52.1 ± 8.4∗↑	64.0 ± 7.7∗↑	71.9 ± 3.3∗↑	51.9 ± 3.6∗↑	60.2 ± 4.4∗↑
GLCM	51.5 ± 7.3∗	17.0 ± 8.6∗↑	33.6 ± 6.1∗↑	48.9 ± 7.7∗↑	53.6 ± 7.0∗↑	38.7 ± 6.5∗↑	44.6 ± 6.3∗↑	48.0 ± 5.1∗↑	43.9 ± 6.2∗↑	47.3 ± 6.3∗↑
$LBP^{riu2}_{24,3}$	66.7 ± 2.2∗	22.0 ± 6.6∗↑	37.4 ± 2.8∗↑	60.9 ± 2.3∗↑	69.6 ± 2.8∗↑	36.8 ± 2.8∗↑	46.2 ± 4.2∗↑	65.0 ± 3.0∗↑	48.4 ± 2.7∗↑	61.7 ± 3.2∗↑
$CLBP_{24,3}$	83.7 ± 3.7∗	36.5 ± 4.9∗↑	33.7 ± 3.8∗↑	79.5 ± 3.6∗↑	**83.6 ± 3.2**∗	69.7 ± 5.8∗↑	76.6 ± 3.6∗↑	84.1 ± 4.0∗	60.9 ± 1.8∗↑	69.0 ± 5.1∗
$LBC_{24,3}$	64.1 ± 3.0∗	22.6 ± 6.1∗↑	36.2 ± 3.7∗↑	58.2 ± 2.4∗↑	67.7 ± 2.7∗↑	33.4 ± 6.1∗↑	41.6 ± 5.5∗↑	60.8 ± 2.6∗↑	46.6 ± 1.9∗↑	60.3 ± 4.1∗↑
$CLBC_{24,3}$	78.6 ± 4.0∗	35.4 ± 3.0∗↑	34.9 ± 5.0∗↑	78.1 ± 3.8∗↑	79.4 ± 4.4∗↑	66.9 ± 7.0∗↑	74.9 ± 3.2∗↑	79.4 ± 4.0∗↑	59.3 ± 1.5∗↑	60.1 ± 5.8∗↑
GP-criptor^{ri}	**90.9 ± 1.9**	**60.9 ± 3.1**∗	**50.6 ± 1.2**∗	**85.4 ± 1.4**∗	82.4 ± 1.9∗	**79.9 ± 2.6**∗	**82.2 ± 1.4**∗	**85.7 ± 1.5**∗	**69.9 ± 1.7**∗	**70.5 ± 2.3**∗
	BrWiRo									
DIF	36.5 ± 2.9∗	14.8 ± 4.1∗↑	30.9 ± 4.1∗↑	34.9 ± 2.5∗↑	32.5 ± 2.4∗↑	21.2 ± 5.8∗↑	34.1 ± 4.1∗↑	34.2 ± 4.3∗↑	34.1 ± 3.6∗↑	34.1 ± 3.4∗↑
$LBP^{u2}_{8,1}$	37.6 ± 1.4∗	23.7 ± 2.3∗↑	18.9 ± 3.0∗↑	31.1 ± 1.5∗↑	36.3 ± 1.9∗↑	24.6 ± 4.8∗↑	37.7 ± 5.0∗↑	40.6 ± 1.9∗↑	30.1 ± 1.8∗↑	33.5 ± 3.0∗↑
GLCM	41.1 ± 6.4∗	13.9 ± 5.3∗↑	27.8 ± 4.7∗↑	39.9 ± 7.3∗↑	38.7 ± 4.6∗↑	29.1 ± 5.6∗↑	39.6 ± 4.6∗↑	37.9 ± 5.0∗↑	35.3 ± 4.5∗↑	38.2 ± 6.0∗↑
$LBP^{riu2}_{24,3}$	68.3 ± 3.4∗	21.8 ± 9.5∗↑	41.4 ± 2.3∗↑	62.7 ± 3.6∗↑	68.5 ± 2.7∗↑	40.1 ± 4.9∗↑	50.8 ± 3.4∗↑	66.6 ± 1.9∗↑	51.5 ± 1.8∗↑	62.2 ± 4.3∗↑
$CLBP_{24,3}$	86.1 ± 3.1∗	38.6 ± 4.3∗↑	33.8 ± 3.8∗↑	78.9 ± 3.5∗↑	**86.3 ± 3.3**∗	73.9 ± 2.5∗↑	79.4 ± 2.8∗↑	85.3 ± 2.8∗	62.1 ± 1.6∗↑	**72.4 ± 6.1**∗
$LBC_{24,3}$	64.0 ± 3.5∗	22.7 ± 8.3∗↑	36.9 ± 3.9∗↑	59.5 ± 4.0∗↑	65.1 ± 2.3∗↑	35.3 ± 4.3∗↑	47.0 ± 2.3∗↑	62.5 ± 2.5∗↑	48.0 ± 1.9∗↑	57.7 ± 3.3∗↑
$CLBC_{24,3}$	84.9 ± 2.7∗	36.5 ± 6.4∗↑	36.7 ± 4.1∗↑	82.9 ± 3.0∗↑	84.6 ± 2.4∗	75.0 ± 3.4∗↑	78.7 ± 3.7∗↑	84.1 ± 2.1∗↑	62.3 ± 1.2∗↑	68.7 ± 6.3∗
GP-criptor^{ri}	**92.5 ± 1.1**	**59.7 ± 3.4**∗	**48.9 ± 0.9**∗	**86.7 ± 1.5**∗	83.7 ± 1.4∗	**81.1 ± 3.0**∗	**83.2 ± 0.9**∗	**86.5 ± 2.0**∗	**69.6 ± 1.2**∗	71.3 ± 1.8∗
	KyNoRo									
DIF	22.2 ± 1.7∗	10.5 ± 2.7∗↑	28.4 ± 1.8∗↑	18.7 ± 1.1∗↑	23.0 ± 2.0∗↑	19.4 ± 2.8∗↑	20.0 ± 2.6∗↑	27.1 ± 2.5∗↑	22.4 ± 0.9∗↑	21.1 ± 1.8∗↑
$LBP^{u2}_{8,1}$	62.9 ± 4.2∗	31.1 ± 7.4∗↑	29.2 ± 3.3∗↑	67.2 ± 4.0∗↑	63.8 ± 3.2∗↑	62.3 ± 4.0∗↑	61.6 ± 4.3∗↑	65.0 ± 3.5∗↑	54.8 ± 1.6∗↑	57.8 ± 4.1∗↑
GLCM	68.0 ± 3.0∗	18.2 ± 3.9∗↑	**44.5 ± 5.0**∗	68.9 ± 1.8∗↑	64.5 ± 3.7∗↑	48.1 ± 5.7∗↑	55.1 ± 3.1∗↑	65.7 ± 3.8∗↑	56.0 ± 2.1∗↑	67.0 ± 2.8∗
$LBP^{riu2}_{24,3}$	68.2 ± 2.7∗	21.5 ± 4.7∗↑	43.5 ± 2.7∗↓	67.0 ± 2.5∗↑	67.9 ± 2.0∗↑	40.9 ± 4.1∗↑	51.6 ± 3.0∗↑	65.3 ± 3.2∗↑	55.1 ± 2.2∗↑	65.9 ± 2.1∗
$CLBP_{24,3}$	90.3 ± 0.7−	43.5 ± 1.9∗↑	34.9 ± 3.1∗↑	40.1 ± 2.8∗↑	90.3 ± 1.2−↓	66.6 ± 14.4∗	**78.7 ± 3.3**∗↓	90.1 ± 1.4−↓	63.7 ± 0.8∗	78.5 ± 4.5∗↓
$LBC_{24,3}$	67.5 ± 3.0∗	17.8 ± 2.7∗↑	41.0 ± 4.1∗	65.4 ± 2.8∗↑	69.2 ± 2.7∗↑	44.5 ± 4.1∗↑	53.0 ± 3.6∗↑	65.6 ± 2.9∗↑	53.5 ± 1.7∗↑	64.8 ± 3.4∗
$CLBC_{24,3}$	**91.1 ± 1.1**−	41.4 ± 3.0∗↑	36.3 ± 4.2∗↑	72.0 ± 2.7∗↑	**90.7 ± 1.3**−↓	66.8 ± 14.3∗	76.3 ± 3.4∗	**91.1 ± 1.3**−↓	**64.5 ± 1.3**∗	**79.2 ± 3.7**∗↓
GP-criptor^{ri}	86.7 ± 1.8	**56.8 ± 2.5**∗	41.1 ± 1.3∗	**82.7 ± 1.8**∗	78.8 ± 1.7∗	**70.7 ± 4.2**∗	75.1 ± 1.6∗	82.6 ± 2.1∗	64.4 ± 1.2∗	66.9 ± 1.7∗
	KyWiRo									
DIF	23.0 ± 0.9∗	12.8 ± 2.3∗↑	28.5 ± 2.9∗↑	19.8 ± 0.9∗↑	20.0 ± 1.6∗↑	21.9 ± 0.9∗↑	22.1 ± 1.4∗↑	22.0 ± 0.1∗↑	23.3 ± 0.5∗↑	17.7 ± 2.7∗↑
$LBP^{u2}_{8,1}$	41.0 ± 0.6∗	21.2 ± 1.9∗↑	23.7 ± 2.2∗↑	37.7 ± 0.6∗↑	36.3 ± 1.1∗↑	44.1 ± 1.9∗↑	46.4 ± 3.8∗↑	40.7 ± 1.4∗↑	37.2 ± 0.4∗↑	32.5 ± 1.3∗↑
GLCM	49.0 ± 0.4∗	16.7 ± 2.8∗↑	32.8 ± 2.1∗↑	49.8 ± 0.7∗↑	47.3 ± 0.3∗↑	33.8 ± 2.8∗↑	41.8 ± 2.0∗↑	46.1 ± 0.4∗↑	40.0 ± 0.9∗↑	47.0 ± 1.1∗↑
$LBP^{riu2}_{24,3}$	69.1 ± 2.2∗	25.0 ± 7.3∗↑	**43.3 ± 4.8**∗	66.4 ± 2.6∗↑	67.1 ± 2.8∗↑	42.9 ± 7.4∗↑	51.6 ± 3.9∗↑	67.2 ± 3.0∗↑	54.7 ± 2.5∗↑	66.9 ± 3.5∗
$CLBP_{24,3}$	90.6 ± 2.2−	38.9 ± 2.1∗↑	34.2 ± 2.6∗↑	39.2 ± 2.3∗↑	**91.4 ± 1.9**−↓	72.4 ± 5.1∗	77.8 ± 6.5∗	90.6 ± 2.4−↓	62.2 ± 1.1∗↑	77.9 ± 5.4∗↓
$LBC_{24,3}$	68.2 ± 1.9∗	17.7 ± 3.4∗↑	39.2 ± 4.0∗	65.1 ± 2.4∗↑	65.9 ± 3.0∗↑	42.7 ± 6.8∗↑	50.7 ± 3.6∗↑	66.2 ± 2.7∗↑	53.0 ± 2.0∗↑	65.6 ± 2.7∗↑
$CLBC_{24,3}$	**91.0 ± 2.1**−	37.9 ± 4.9∗↑	36.3 ± 4.8∗↑	71.4 ± 3.4∗↑	91.2 ± 2.2−↓	**73.1 ± 4.5**∗	**78.6 ± 5.3**∗	**91.2 ± 2.7**−↓	63.8 ± 1.3∗↑	**79.9 ± 4.1**∗↓
GP-criptor^{ri}	88.5 ± 1.4	**56.2 ± 1.4**∗	41.1 ± 1.7∗	**84.4 ± 1.8**∗	80.5 ± 1.2∗	72.7 ± 3.8∗	76.3 ± 1.6∗	84.0 ± 1.4∗	**65.3 ± 1.1**∗	68.3 ± 1.8∗
	OutexTC00									
DIF	17.4 ± 1.7∗	9.0 ± 3.3∗↑	13.2 ± 4.7∗↑	14.0 ± 3.0∗↑	15.4 ± 2.1∗↑	11.1 ± 2.8∗↑	12.3 ± 4.1∗↑	15.7 ± 2.2∗↑	13.0 ± 1.5∗↑	14.7 ± 2.9∗↑
$LBP^{u2}_{8,1}$	87.6 ± 3.3	45.5 ± 8.7∗↑	33.1 ± 8.8∗↑	85.5 ± 2.5∗	**84.9 ± 3.0**∗	75.8 ± 5.8∗	76.1 ± 4.2∗↑	**89.3 ± 1.6** ↓	65.5 ± 6.2∗↑	**80.3 ± 7.2**∗↓
GLCM	70.5 ± 2.4∗	20.6 ± 7.2∗↑	35.9 ± 7.4∗↑	59.8 ± 5.6∗↑	65.7 ± 4.8∗↑	39.5 ± 13.0∗↑	50.8 ± 7.6∗↑	69.0 ± 3.1∗↑	49.1 ± 5.5∗↑	69.9 ± 2.8∗↑
$LBP_{24,3}$	69.3 ± 3.6∗	29.4 ± 9.8∗↑	39.3 ± 4.4∗↑	66.5 ± 4.1∗↑	64.9 ± 2.6∗↑	39.0 ± 5.1∗↑	50.7 ± 3.9∗↑	66.7 ± 4.3∗↑	52.2 ± 2.1∗↑	68.9 ± 2.7∗↑
$CLBP_{24,3}$	79.3 ± 2.9∗	42.8 ± 5.9∗↑	26.5 ± 3.4∗↑	27.8 ± 1.5∗↑	81.3 ± 1.9∗↑	63.8 ± 5.2∗↑	67.5 ± 3.7∗↑	80.3 ± 2.0∗↑	55.7 ± 1.4∗↑	69.4 ± 3.3∗↑
$LBC_{24,3}$	67.8 ± 3.0∗	22.3 ± 5.0∗↑	34.8 ± 5.8∗↑	66.1 ± 3.0∗↑	63.6 ± 3.1∗↑	37.5 ± 7.6∗↑	46.2 ± 5.0∗↑	65.4 ± 3.5∗↑	49.9 ± 2.0∗↑	66.5 ± 2.0∗↑
$CLBC_{24,3}$	77.9 ± 2.5∗	34.5 ± 12.1∗↑	27.7 ± 2.8∗↑	62.5 ± 2.1∗↑	79.6 ± 1.8∗↑	65.5 ± 3.7∗↑	65.6 ± 4.3∗↑	78.5 ± 1.3∗↑	57.4 ± 1.4∗↑	71.3 ± 4.2∗
GP-criptor^{ri}	**87.7 ± 1.9**	**57.1 ± 2.7**∗	**47.3 ± 2.2**∗	**87.6 ± 2.4**	83.8 ± 1.3∗	**72.0 ± 3.8**∗	**79.9 ± 1.4**∗	85.4 ± 2.2∗	**72.6 ± 1.1**∗	73.9 ± 1.6∗
	OutexTC10									
DIF	8.2 ± 0.7∗	6.6 ± 1.5∗↑	11.4 ± 2.1∗↑	7.4 ± 0.7∗↑	8.4 ± 0.7∗↑	7.5 ± 1.6∗↑	7.7 ± 1.7∗↑	9.4 ± 0.7∗↑	9.1 ± 0.8∗↑	7.8 ± 1.2∗↑
$LBP^{u2}_{8,1}$	37.4 ± 2.3∗	18.9 ± 3.0∗↑	27.6 ± 4.0∗↑	34.3 ± 1.7∗↑	34.4 ± 2.9∗↑	31.6 ± 3.1∗↑	42.1 ± 3.9∗↑	36.1 ± 2.4∗↑	27.4 ± 0.8∗↑	32.7 ± 2.6∗↑
GLCM	43.8 ± 2.5∗	17.3 ± 4.8∗↑	26.4 ± 6.5∗↑	41.2 ± 6.2∗↑	36.5 ± 3.3∗↑	28.0 ± 4.9∗↑	38.0 ± 3.8∗↑	43.3 ± 3.6∗↑	34.4 ± 3.9∗↑	44.2 ± 3.8∗↑
$LBP^{riu2}_{24,3}$	67.6 ± 2.0∗	21.9 ± 6.9∗↑	36.4 ± 3.9∗↑	66.4 ± 1.1∗↑	66.0 ± 2.5∗↑	38.8 ± 6.1∗↑	49.8 ± 2.7∗↑	63.6 ± 4.0∗↑	48.2 ± 2.1∗↑	64.1 ± 2.4∗↑
$CLBP_{24,3}$	85.4 ± 2.2	34.1 ± 7.1∗↑	26.6 ± 4.7∗↑	15.5 ± 2.3∗↑	85.8 ± 2.0 ↓	58.1 ± 10.0∗↑	72.0 ± 6.2∗↑	85.4 ± 2.3	58.2 ± 1.2∗↑	72.2 ± 4.0∗
$LBC_{24,3}$	63.8 ± 2.0∗	19.6 ± 4.4∗↑	34.7 ± 3.8∗↑	62.6 ± 2.0∗↑	60.9 ± 2.9∗↑	37.6 ± 6.3∗↑	47.0 ± 3.3∗↑	58.9 ± 4.1∗↑	46.1 ± 1.3∗↑	59.0 ± 1.5∗↑
$CLBC_{24,3}$	86.2 ± 2.1	40.9 ± 6.1∗↑	28.2 ± 4.6∗↑	70.4 ± 2.3∗↑	**86.8 ± 1.8** ↓	63.5 ± 7.1∗↑	74.3 ± 3.5∗↑	**86.4 ± 2.2**	61.1 ± 1.2∗↑	**77.2 ± 5.2**∗↓
GP-criptor^{ri}	**86.8 ± 1.9**	**51.2 ± 3.1**∗	**41.5 ± 2.0**∗	**86.2 ± 1.6**	83.4 ± 1.5∗	**70.5 ± 4.6**∗	**78.1 ± 1.9**∗	85.6 ± 2.1	**68.6 ± 1.7**∗	72.2 ± 1.1∗

VI. FURTHER ANALYSIS

In this section, a comparison of the robustness of GP-criptor (the baseline method) and GP-criptor^{ri} (the newly proposed method) to evolve rotation-invariant image descriptors is discussed first. Then an individual program evolved by GP-criptor^{ri} is analysed and discussed in order to provide more understanding on why and how the proposed method can perform well.

A. Comparison between GP-criptor and GP-criptor^{ri}

To support our hypothesis of the inability of GP-criptor [44] (the baseline method) to handle the rotation variance, we have experimented with this method using the same six texture image datasets discussed in Section IV-A. The results of the non-rotated and rotated versions for each dataset (Brodatz, Kylberg and OutexTC) are grouped in a single table in this section. A **bold face** font is used in those tables to present the results of statistical testing (significantly better).

The results of using the 10 previously used classification methods (Section V) with GP-criptor and GP-criptor^{ri} evolved descriptors on the BrNoRo and BrWiRo datasets are presented in Table V. Apart from AdaBoost and J48, all other classifiers have achieved significantly better performance on BrNoRo (fixed in terms of rotation) using the descriptors evolved

TABLE V
ACCURACY (%) OF 10 CLASSIFIERS USING GP-CRIPTOR AND GP-CRIPTOR^{ri} EVOLVED IMAGE DESCRIPTORS ON THE BRNORO AND BRWIRO DATASETS (MEAN ± STANDARD DEVIATION).

	BrNoRo		BrWiRo	
	GP-criptor	GP-criptor^{ri}	GP-criptor	GP-criptor^{ri}
1NN	**96.3 ± 0.8**	90.9 ± 1.9	60.3 ± 1.7	**92.5 ± 1.1**
AdaBoost	48.9 ± 2.7	**60.9 ± 3.1**	26.7 ± 0.8	**59.7 ± 3.4**
J48	39.1 ± 1.0	**50.6 ± 1.2**	25.9 ± 1.3	**48.9 ± 0.9**
K*	**95.1 ± 0.4**	85.4 ± 1.4	51.8 ± 1.9	**86.7 ± 1.5**
MLP	**94.4 ± 0.9**	82.4 ± 1.9	52.6 ± 2.1	**83.7 ± 1.4**
NB	**89.3 ± 1.1**	79.9 ± 2.6	44.1 ± 3.7	**81.1 ± 3.0**
NBTree	**84.5 ± 1.0**	82.2 ± 1.4	50.4 ± 2.0	**83.2 ± 0.9**
NNge	**95.1 ± 0.7**	85.7 ± 1.5	56.3 ± 2.4	**86.5 ± 2.0**
RF	**71.1 ± 2.0**	69.9 ± 1.7	39.9 ± 0.9	**69.6 ± 1.2**
SVM	**81.0 ± 1.2**	70.5 ± 2.3	42.9 ± 2.1	**71.3 ± 1.8**

TABLE VI
ACCURACY (%) OF 10 CLASSIFIERS USING GP-CRIPTOR AND GP-CRIPTOR^{ri} EVOLVED IMAGE DESCRIPTORS ON THE KYNORO AND KYWIRO DATASETS (MEAN ± STANDARD DEVIATION).

	KyNoRo		KyWiRo	
	GP-criptor	GP-criptor^{ri}	GP-criptor	GP-criptor^{ri}
1NN	**89.5 ± 1.3**	86.7 ± 1.8	56.3 ± 2.4	**88.5 ± 1.4**
AdaBoost	44.0 ± 1.2	**56.8 ± 2.5**	23.5 ± 1.1	**56.2 ± 1.4**
J48	33.9 ± 1.1	**41.1 ± 1.3**	25.1 ± 1.6	**41.1 ± 1.7**
K*	**85.9 ± 1.5**	82.7 ± 1.8	50.5 ± 2.0	**84.4 ± 1.8**
MLP	**82.5 ± 2.0**	78.8 ± 1.7	43.1 ± 1.5	**80.5 ± 1.2**
NB	72.9 ± 5.4	70.7 ± 4.2	38.4 ± 2.2	**72.7 ± 3.8**
NBTree	75.8 ± 1.7	75.1 ± 1.6	46.7 ± 1.7	**76.3 ± 1.6**
NNge	**85.4 ± 1.7**	82.6 ± 2.1	49.4 ± 2.1	**84.0 ± 1.4**
RF	64.5 ± 1.4	64.4 ± 1.2	38.7 ± 1.1	**65.3 ± 1.1**
SVM	**70.5 ± 2.0**	66.9 ± 1.7	37.0 ± 1.4	**68.3 ± 1.8**

by the baseline method than those were evolved by GP-criptor^{ri}. However, AdaBoost and J48 show, respectively, 12% and 11.5% improvement using GP-criptor^{ri} descriptors. More importantly the proposed method shows significantly positive influence on the performances of *all* the 10 classifiers on BrWiRo (with rotation) compared to the baseline method, where the improvement is ranging on average between 37% (NB) and 23% (J48).

Table VI presents the results of these two methods' (GR-criptor and GP-criptor^{ri}) evolved descriptors for the 10 classification methods on the KyNoRo and KyWiRo datasets. Although the classifiers showed better performance on the non-rotated version of the Kylberg dataset (KyNoRo) using the features extracted by the baseline method descriptors, GP-criptor^{ri} has achieved comparable performance and the gap is only 3.7% in the worst case (MLP). Noticeably, GP-criptor^{ri} has improved the performances of AdaBoost and J48 by 12.8% and 7.2%, respectively. The results on the rotated version of Kylberg (KyWiRo) show the ability of the new method to handle the rotation variance where the baseline method has struggled to preserve the same level of performance on the rotation-free version of this dataset.

Finally, the results presented in Table VII are obtained on the OutexTC00 and OutexTC10 datasets. The pattern of the results on these two datasets is very similar to that previously observed on the Kylberg and Brodatz datasets. The baseline method shows significantly better performance than that achieved by the new method on the rotation-free version of OutexTC (OutexTC00), whereas the new method shows significantly better performance on the rotated version, revealing the robustness of this method to evolve rotation-invariant image descriptor. The differences between the performances of the two methods are ranging between 22.3% (NB) and 12.1% (J48).

TABLE VII
ACCURACY (%) OF 10 CLASSIFIERS USING GP-CRIPTOR AND GP-CRIPTOR^{ri} EVOLVED IMAGE DESCRIPTORS ON THE OUTEXTC00 AND OUTEXTC10 DATASETS (MEAN ± STANDARD DEVIATION).

	OutexTC00		OutexTC10	
	GP-criptor	GP-criptor^{ri}	GP-criptor	GP-criptor^{ri}
1NN	**95.9 ± 2.0**	87.7 ± 1.9	69.3 ± 2.7	**86.8 ± 1.9**
AdaBoost	55.5 ± 1.2	57.1 ± 2.7	30.5 ± 1.0	**51.2 ± 3.1**
J48	45.9 ± 2.9	47.3 ± 2.2	29.4 ± 2.2	**41.5 ± 2.0**
K*	**93.4 ± 2.4**	87.6 ± 2.4	65.5 ± 2.8	**86.2 ± 1.6**
MLP	**95.8 ± 1.5**	83.8 ± 1.3	66.5 ± 3.1	**83.4 ± 1.5**
NB	**86.7 ± 3.0**	72.0 ± 3.8	48.2 ± 2.9	**70.5 ± 4.6**
NBTree	**85.7 ± 2.0**	79.9 ± 1.4	58.2 ± 2.4	**78.1 ± 1.9**
NNge	**95.4 ± 2.4**	85.4 ± 2.2	66.8 ± 2.9	**85.6 ± 2.1**
RF	**78.4 ± 1.1**	72.6 ± 1.1	47.4 ± 2.4	**68.6 ± 1.7**
SVM	**84.8 ± 1.6**	73.9 ± 1.6	54.9 ± 1.9	**72.2 ± 1.1**

In summary, the following observations can be made:

- Although the baseline method has better performance on the rotation-free datasets, the new method has achieved comparable performance to GP-criptor and still outperformed the other image descriptors studied in Section V;
- The new method has the potential to handle the rotation variance, whilst the baseline method struggled to preserve a satisfactory level of performance when the dataset has rotated instances;
- The proposed method achieved more stable or consistent results between the rotated and rotation-free datasets, compared to those results achieved by the baseline method as the drop in performance was significant on the rotated images; and
- The proposed method has a noticeable positive influence on improving the performances of AdaBoost and J48 classification methods.

B. Analysis of a GP-criptor^{ri} evolved descriptor

Here, we have chosen an individual that has been evolved on the BrWiRo dataset. The tree representation of the program is depicted in Fig. 14. This program uses a 7-bit code length and a sliding window of size 5×5 pixels. Overall, there are 106 nodes in this program with 56 terminals and 50 functions. Hence, this program performs on average 7 operations $((50 - 1\ (\text{code}))\,/7 = 7)$ to calculate the value of each bit. Most of these tree branches can be interpreted easily such as the second $(min + 2\,(mean - max))$, third $(mean^2 - ((min \times max)))$, fourth $((2 \times mean) - max)$, and the simplified sixth $(min^2 - stdev)$ bit branches; whereas other branches are more complicated.

In terms of accuracy, this program has achieved 94.6% accuracy on the unseen data; the confusion matrix (where the row indices represent the actual classes, while column indices the predicted classes) is presented in Table VIII. The program has successfully classified all instances (100%) of 6 out of the 20 classes, over 90% accuracy on the other 11 classes, and only 3 classes are below 80%.

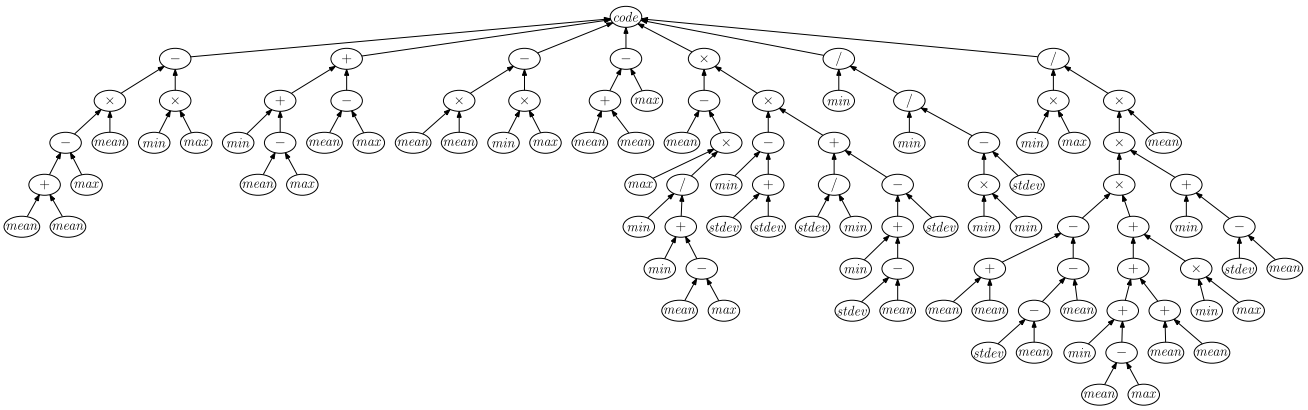


Fig. 14. The tree representation of a program evolved on the BrWiRo dataset.

TABLE VIII
THE CONFUSION MATRIX FOR THE PROGRAM PRESENTED IN FIG. 14

	D01	D03	D04	D05	D06	D09	D11	D14	D15	D16	D17	D18	D20	D21	D24	D34	D37	D46	D47	D49	
D01	504	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.0
D03	0	466	0	0	0	0	0	0	14	0	0	0	0	0	24	0	0	0	0	0	92.5
D04	0	0	389	0	0	63	52	0	0	0	0	0	0	0	0	0	0	0	0	0	77.2
D05	0	0	1	493	0	0	0	0	5	0	0	5	0	0	0	0	0	0	0	0	97.8
D06	0	0	0	0	504	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.0
D09	0	0	87	0	0	385	3	0	7	0	1	0	0	0	21	0	0	0	0	0	76.4
D11	0	0	18	0	0	4	482	0	0	0	0	0	0	0	0	0	0	0	0	0	95.6
D14	0	0	0	0	0	0	0	501	1	0	0	0	0	0	2	0	0	0	0	0	99.4
D15	0	21	10	2	0	6	0	0	451	0	0	0	0	0	14	0	0	0	0	0	89.5
D16	0	0	0	0	0	0	2	0	0	489	13	0	0	0	0	0	0	0	0	0	97.0
D17	0	0	2	0	0	0	0	0	0	0	502	0	0	0	0	0	0	0	0	0	99.6
D18	0	0	0	109	0	0	0	0	0	0	0	395	0	0	0	0	0	0	0	0	78.4
D20	0	0	0	0	0	0	0	0	0	0	0	0	504	0	0	0	0	0	0	0	100.0
D21	0	0	0	0	0	0	0	0	0	0	5	0	0	499	0	0	0	0	0	0	99.0
D24	0	3	1	0	0	2	0	0	0	0	0	0	0	0	498	0	0	0	0	0	98.8
D34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	504	0	0	0	0	100.0
D37	2	0	2	0	0	0	1	0	0	0	0	20	0	0	0	0	478	0	1	0	94.8
D46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	504	0	0	100.0
D47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	504	0	100.0
D49	0	12	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	488	96.8

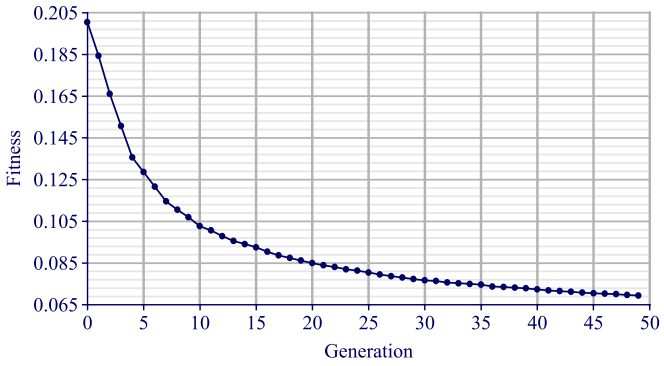


Fig. 15. The average fitness value per generation.

The average fitness value at each generation of the 30 runs, using the same training set that was used to evolve the program in Fig. 14, is presented in Fig. 15. Clearly, the system has made fast jumps in the fitness values over the first 16 generations as the fitness value has decreased from approximately 0.2 to approximately 0.089; while the progress after that was slower and the fitness value dropped to approximately 0.067 over the remaining 34 generations.

In order to shed light on the time required to evolve a descriptor by GP-criptor^{ri}, the CPU time for each evolutionary run has been measured from the beginning of generating the initial population to the end when a stopping criterion is met. Fig. 16 presents the average time required to evolve an image

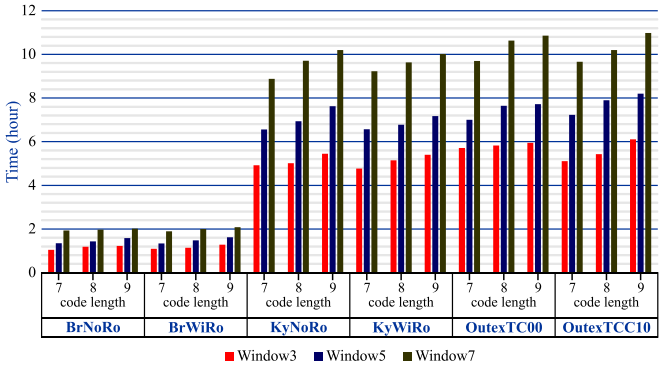


Fig. 16. The average time in hours required to evolve a descriptor.

descriptor by GP-criptor^{ri} for different window sizes and code lengths. As expected, the larger the window size, the longer time that is needed to obtain a good descriptor.

A stacked representation of the resulting feature vectors by this program for 40 instances, 2 from each class, that were randomly drawn from the BrWiRo dataset is presented in Fig. 17. The class labels for those instances are printed on the horizontal axis, whereas the vertical axis shows the relative frequency. The aim of this figure is to show that a program evolved by GP-criptor^{ri} is responding in a similar way to those instances belonging to the same class, and differently to those instances belonging to other classes. Each feature vector comprises 128 values (2^7), each of which has been represented with a different colour in the bars of Fig. 17. Hence, each colour indicates the same exact feature across those bars. A closer inspection of this figure reveals how those feature vectors belonging to the same class are similar and have, to some extent, a distinctive fingerprint. Some typical examples are D09, D17, D24, and D49 as they are visually easier to compare than others. The figure also shows two important facts: firstly, the system has detected some keypoints that appeared in one class but not, or very seldom, in the other classes; and secondly, the system is able to find keypoints that are shared between all those classes but appear more frequently in one class than in the other classes. An example of the former case is the feature indicated in dark-blue at the middle part of class D34, whereas the features indicated by

light-blue and light-green of, respectively, classes D21 and D49 are examples of the latter case.

Further to our earlier discussion on the number of training instances (Section I), and from the example studied in this section, it becomes evident that the system did not stop the evolutionary process when only one or a few distinctive keypoints have been detected. Instead, the proposed method continued the process of evolving a better program that can capture as many prominent keypoints as possible. The main idea of using only a few instances matches reasonably well the process of how humans teach young children to identify different objects. There are three amazing facts in this learning process: (1) humans do not need to use thousands or hundreds of images where only one or a few are enough; (2) different children may detect or use different characteristics to identify the object of each category, e.g., some may focus on the body shape while others may focus on the head/face characteristics in the case of discriminating between cow and horse examples; and (3) humans do not specify keypoints for objects of different categories and teach the children to use them to perform the categorisation task, instead, the children identify those keypoints themselves. Fig. 17 shows that the evolved program (presented in Fig. 14) captured and extracted similar patterns for the two instances of each class at the end of the evolutionary process, but the patterns for different classes are distinguished. This shows that this method is very good at distinguishing examples in different classes.

VII. Conclusions

In this paper, a GP approach was proposed to automatically evolve rotation-invariant image descriptors to detect good keypoints and extract informative features simultaneously for texture image classification. Different from existing methods, the proposed GP approach does not require any human intervention, needs only two instances per class, and aims to tackle rotation (in)variance by using simple rotation-invariant features in the terminal set. This method is suitable for problems where only a small number of labelled instances are available, and for the situations that cannot afford a long time for training. To examine the performance of the proposed GP approach, a large number of experiments have been conducted on six texture image classification datasets of varying difficulty, with different degrees of rotations. The performance of the GP approach is compared with seven state-of-the-art domain-expert designed image descriptors and ten well-known classifiers are used in the experiments. The results show that the proposed GP method, using only two instances per class, performed comparably or significantly outperformed the other methods in most cases. Furthermore, the GP approach is robust in handling different degrees of rotations, and the evolved GP tree, i.e., image descriptor, is understandable and interpretable by humans, although it is automatically constructed without human intervention.

A. Major Contributions

This paper brings the following major contributions. Firstly, it shows that, by carefully designing the program structure, GP can automatically detect good keypoints and extract informative features simultaneously in a single process for texture image classification. By using simple pixel statistics, such as min, max, mean, standard deviation and simple arithmetic operators as terminals and functions in the program representation, the proposed GP system can effectively solve texture classification tasks with different rotations/orientations. Secondly, this paper shows how GP can effectively construct and generate good image descriptors using only two instances per class in the training set as inputs. Rather than using the classification accuracy or error rate that are commonly used in GP and many other learning systems for classification, this paper develops a new measure that can effectively use only two instances per class to evolve good image descriptors. The image descriptors automatically constructed by GP outperform state-of-the-art image descriptors carefully designed by domain-experts on six image datasets with different rotations. Thirdly, the proposed GP method can automatically extract more informative features than those obtained from domain-expert designed descriptors; the extracted features are not biased to any specific classifier and can be used as inputs to different types of learning/classification methods, achieving good performance.

B. Future Work

Although the proposed method has been shown to be a promising alternative to those domain-expert designed methods, it has some limitations that we intend to address in the future. For example, the window size and code length parameters are currently required to be empirically determined. This could be tackled via changing the program representation to handle those parameters side-by-side with the process of evolving a descriptor. Moreover, the method has been assessed using only texture image datasets. Hence, in the future we would like to investigate the ability of the proposed method for more complex classification tasks, e.g., involving general or real-life, non-texture images. Performing classification on real-life image problems is difficult as it is more likely that the object of a category will appear on different backgrounds, which may require adding more instances during the learning phase. Transfer learning [82], [83], where examples of a related (source) domain can be used to evolve a model that can be adopted to the target domain, can be applied where only a few instances are available. Dealing with scale variation is a more challenging task and we would like to extend the proposed method to handle this challenge in the future. Another important factor that we would like to study is the influence of using more than two instances per class on the performance of the proposed method.

References

[1] B. Bhanu, L. Yingqiang, and K. Krawiec, *Evolutionary Synthesis of Pattern Recognition Systems*, ser. Monographs in Computer Science. Springer, 2006.

[2] L. Wang, B. Yang, S. Wang, and Z. Liang, "Building image feature kinetics for cement hydration using gene expression programming with similarity weight tournament selection," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 679–693, 2015.

[3] G. Karafotias, M. Hoogendoorn, and A. Eiben, "Parameter control in evolutionary algorithms: Trends and challenges," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 167–187, 2015.

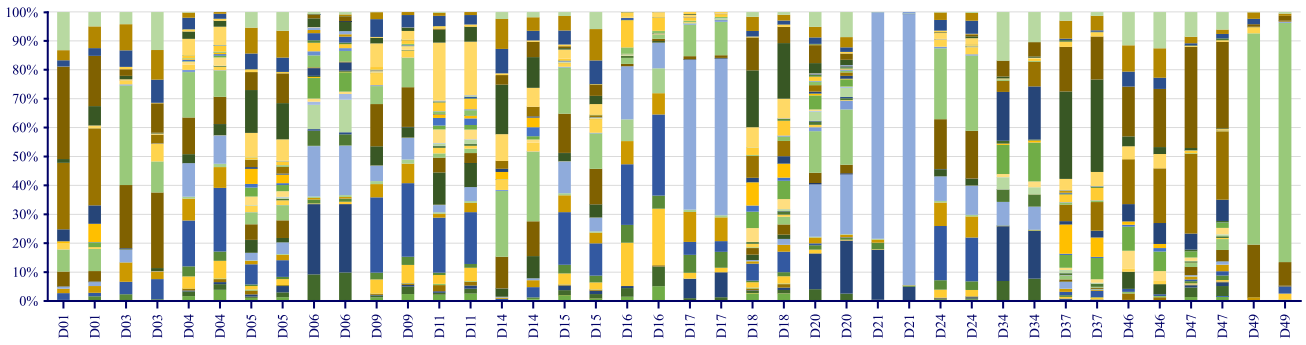


Fig. 17. A stacked representation of the resulting feature vectors by the program shown in Fig. 14 for 40 instances (2 from each class) drawn from the BrWiRo. The same features are indicated using the same colours to highlight the similarities/differences between instances of the same/different class(es).

[4] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*. Alvey Vision Club, 1988, pp. 147–151.

[5] M. Zhang, V. Ciesielski, and P. Andreae, "A domain-independent window approach to multiclass object detection using genetic programming," *EURASIP Journal on Advances in Signal Processing*, vol. 2003, no. 8, pp. 841–859, 2003.

[6] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R*. Springer, 2014.

[7] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 3, pp. 177–280, 2008.

[8] G. Carneiro and A. Jepson, "The distinctiveness, detectability, and robustness of local image features," in *Proceedings of 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2. IEEE, 2005, pp. 296–301.

[9] K. Juneja, A. Verma, and S. Goel, "A survey on recent image indexing and retrieval techniques for low-level feature extraction in CBIR systems," in *Proceedings of 2015 IEEE International Conference on Computational Intelligence Communication Technology*. IEEE, 2015, pp. 67–72.

[10] R. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-3, no. 6, pp. 610–621, 1973.

[11] T. Ojala, M. Pietikäinen, and D. Harwood, "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions," in *Proceedings of the 12th International Conference on Pattern Recognition*, vol. 1. IEEE, 1994, pp. 582–585.

[12] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision*. IEEE, 1999, pp. 1150–1157.

[13] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[14] J. Chen, S. Shan, C. He, G. Zhao, M. Pietikäinen, X. Chen, and W. Gao, "WLD: A robust local image descriptor," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1705–1720, 2010.

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proceedings of 2011 IEEE International Conference on Computer Vision*. IEEE, 2011, pp. 2564–2571.

[16] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "BRIEF: Computing a local binary descriptor very fast," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1281–1298, 2012.

[17] R. Ortiz, "FREAK: Fast retina keypoint," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2012, pp. 510–517.

[18] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "KAZE features," in *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*. Springer, 2012, pp. 214–227.

[19] B. Yang and S. Chen, "A comparative study on local binary pattern (LBP) based face recognition: LBP histogram versus LBP image," *Neurocomputing*, vol. 120, pp. 365–379, 2013.

[20] M. Lopez-de-la Calleja, T. Nagai, M. Attamimi, M. Nakano-Miyatake, and H. Perez-Meana, "Object detection using SURF and superpixels," *Journal of Software Engineering and Applications*, vol. 6, no. 9, pp. 511–518, 2013.

[21] A. Satpathy, X. Jiang, and H.-L. Eng, "LBP-based edge-texture features for object recognition," *IEEE Transactions on Image Processing*, vol. 23, no. 5, pp. 1953–1964, 2014.

[22] A. Bouganis and M. Shanahan, "Flexible object recognition in cluttered scenes using relative point distribution models," in *Proceedings of the 19th International Conference on Pattern Recognition*. IEEE, 2008, pp. 1–5.

[23] S. Ramamoorthy, R. Kirubakaran, and R. Subramanian, "Texture feature extraction using MGRLBP method for medical image classification," in *Artificial Intelligence and Evolutionary Algorithms in Engineering Systems*, ser. Advances in Intelligent Systems and Computing. Springer, 2015, vol. 324, pp. 747–753.

[24] S. Saxena and R. K. Singh, "A survey of recent and classical image registration methods," *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 7, no. 4, pp. 167–176, 2014.

[25] M.-H. Tsai, Y.-K. Chan, A.-M. Hsu, C.-Y. Chuang, C.-M. Wang, and P.-W. Huang, "Feature-based image segmentation," *Journal of Imaging Science and Technology*, vol. 57, no. 1, pp. 1–12, 2013.

[26] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[27] R. Poli and S. Cagnoni, "Genetic programming with user-driven selection: Experiments on the evolution of algorithms for image enhancement," in *Genetic Programming 1997: Proceedings of the Second Annual Conference*. Morgan Kaufmann, 1997, pp. 269–277.

[28] C. Downey and M. Zhang, "Multiclass object classification for computer vision using linear genetic programming," in *Proceedings of the 24th International Conference on Image and Vision Computing New Zealand*. IEEE, 2009, pp. 73–78.

[29] F. Abdulhamid, K. Neshatian, and M. Zhang, "Image recognition using genetic programming with loop structures," in *Proceedings of the 26th International Conference on Image and Vision Computing New Zealand*, vol. 29, 2011, pp. 553–558.

[30] H. Al-Sahaf, A. Song, K. Neshatian, and M. Zhang, "Extracting image features for classification by two-tier genetic programming," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.

[31] W. Albukhanajer, J. Briffa, and Y. Jin, "Evolutionary multiobjective image feature extraction in the presence of noise," *IEEE Transactions on Cybernetics*, vol. 45, no. 9, pp. 1757–1768, 2015.

[32] A. Song and V. Ciesielski, "Texture segmentation by genetic programming," *Evolutionary Computation*, vol. 16, no. 4, pp. 461–481, 2008.

[33] Y. Liang, M. Zhang, and W. Browne, "Image segmentation: A survey of methods based on evolutionary computation," in *Simulated Evolution and Learning*, ser. Lecture Notes in Computer Science. Springer, 2014, vol. 8886, pp. 847–859.

[34] ——, "A supervised figure-ground segmentation method using genetic programming," in *Proceedings of the 18th European Conference on the Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science, vol. 9028. Springer, 2015, pp. 491–503.

[35] S. Chicotay, O. David, and N. Netanyahu, "Image registration of very large images via genetic programming," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2014, pp. 329–334.

[36] M. Ebner and A. Zell, "Evolving a task specific image operator," in *Evolutionary Image Analysis, Signal Processing and Telecommunications*, ser. Lecture Notes in Computer Science. Springer, 1999, vol. 1596, pp. 74–89.

[37] L. Trujillo and G. Olague, "Synthesis of interest point detectors through genetic programming," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2006, pp. 887–894.

[38] ——, "Automated design of image operators that detect interest points," *Evolutionary Computation*, vol. 16, no. 4, pp. 483–507, 2008.

[39] G. Olague and L. Trujillo, "A genetic programming approach to the design of interest point operators," in *Bio-inspired Hybrid Intelligent Systems for Image Analysis and Pattern Recognition*, ser. Studies in Computational Intelligence. Springer, 2009, vol. 256, pp. 49–65.

[40] C. B. Perez and G. Olague, "Evolutionary learning of local descriptor operators for object recognition," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2009, pp. 1051–1058.

[41] L. Liu, L. Shao, X. Li, and K. Lu, "Learning spatio-temporal representations for action recognition: A genetic programming approach," *IEEE Transactions on Cybernetics*, vol. PP, no. 99, pp. 1–12, 2015, doi:10.1109/TCYB.2015.2399172.

[42] L. Shao, L. Liu, and X. Li, "Feature learning for image classification via multiobjective genetic programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 7, pp. 1359–1371, 2014.

[43] H. Al-Sahaf, A. Song, K. Neshatian, and M. Zhang, "Two-tier genetic programming: Towards raw pixel-based image classification," *Expert Systems with Applications*, vol. 39, no. 16, pp. 12 291–12 301, 2012.

[44] H. Al-Sahaf, M. Zhang, M. Johnston, and B. Verma, "Image descriptor: A genetic programming approach to multiclass texture classification," in *Proceedings of 2015 IEEE Congress on Evolutionary Computation*. IEEE, 2015, pp. 2460–2467.

[45] L. Nanni, S. Brahnam, and A. Lumini, "A simple method for improving local binary patterns by considering non-uniform patterns," *Pattern Recognition*, vol. 45, no. 10, pp. 3844–3852, 2012.

[46] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Gray scale and rotation invariant texture classification with local binary patterns," in *Proceedings of the 6th European Conference on Computer Vision*, ser. Lecture Notes in Computer Science. Springer, 2000, no. 1842, pp. 404–420.

[47] ——, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[48] Z. Guo, L. Zhang, and D. Zhang, "A completed modeling of local binary pattern operator for texture classification," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1657–1663, 2010.

[49] Y. Zhao, D.-S. Huang, and W. Jia, "Completed local binary count for rotation invariant texture classification," *IEEE Transactions on Image Processing*, vol. 21, no. 10, pp. 4492–4497, 2012.

[50] A. Song, T. Loveard, and V. Ciesielski, "Towards genetic programming for texture classification," in *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence*, ser. Lecture Notes in Computer Science, vol. 2256. Springer, 2001, pp. 461–472.

[51] A. Song and V. Ciesielski, "Texture analysis by genetic programming," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE Press, 2004, pp. 2092–2099.

[52] W. A. Tackett, "Genetic programming for feature discovery and image discrimination," in *Proceedings of the 5th International Conference on Genetic Algorithms*, 1993, pp. 303–311.

[53] M. Zhang and V. Ciesielski, "Genetic programming for multiple class object detection," in *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence*, ser. Lecture Notes in Computer Science, vol. 1747. Springer, 1999, pp. 180–192.

[54] T. Loveard and V. Ciesielski, "Representing classification problems in genetic programming," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 2. IEEE, 2001, pp. 1070–1077.

[55] P. Brodatz, *Textures: A Photographic Album for Artists and Designers*. Dover Publications, 1999.

[56] W. R. Smart and M. Zhang, "Classification strategies for image classification in genetic programming," in *Proceedings of the 18th International Conference on Image and Vision Computing New Zealand*. Massey University, 2003, pp. 402–407.

[57] M. Zhang, U. Bhowan, and B. Ny, "Genetic programming for object detection: A two-phase approach with an improved fitness function," *Electronic Letters on Computer Vision and Image Analysis*, vol. 6, no. 1, pp. 27–43, 2007.

[58] M. Zhang and M. Johnston, "A variant program structure in tree-based genetic programming for multiclass object classification," in *Evolutionary Image Analysis and Signal Processing*, ser. Studies in Computational Intelligence. Springer, 2009, vol. 213, pp. 55–72.

[59] D. L. Atkins, K. Neshatian, and M. Zhang, "A domain independent genetic programming approach to automatic feature extraction for image classification," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2011, pp. 238–245.

[60] H. Al-Sahaf, M. Zhang, and M. Johnston, "Genetic programming evolved filters from a small number of instances for multiclass texture classification," in *Proceedings of the 29th International Conference on Image and Vision Computing New Zealand*. ACM, 2014, pp. 84–89.

[61] W. Fu, M. Johnston, and M. Zhang, "Automatic construction of invariant features using genetic programming for edge detection," in *AI 2012: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science. Springer, 2012, vol. 7691, pp. 144–155.

[62] ——, "Distribution-based invariant feature construction using genetic programming for edge detection," *Soft Computing*, vol. 19, no. 8, pp. 2371–2389, 2015.

[63] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[64] A. Kadyrov and M. Petrou, "The trace transform and its applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 8, pp. 811–828, 2001.

[65] D. J. Montana, "Strongly typed genetic programming," *Evolutionary Computation*, vol. 3, no. 2, pp. 199–230, 1995.

[66] G. J. Mclachlan, *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, 2004.

[67] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 1, no. 4, pp. 300–307, 2007.

[68] S. Johnson, *Stephen Johnson on Digital Photography*. O'Reilly Media, Incorporated, 2006.

[69] G. Kylberg, "The Kylberg texture dataset v. 1.0," Centre for Image Analysis, Swedish University of Agricultural Sciences and Uppsala University, Uppsala, Sweden, External report (Blue series) 35, 2011.

[70] T. Ojala, T. Mäenpää, M. Pietikäinen, J. Viertola, J. Kyllonen, and S. Huovinen, "Outex - new framework for empirical evaluation of texture analysis algorithms," in *Proceedings of the 16th International Conference on Pattern Recognition*, vol. 1. IEEE, 2002, pp. 701–706.

[71] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face recognition with local binary patterns," in *Proceedings of the 8th European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, T. Pajdla and J. Matas, Eds., vol. 3021. Springer, 2004, pp. 469–481.

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

[73] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005.

[74] T. Rassem and B. E. Khoo, "Completed local ternary pattern for rotation invariant texture classification," *The Scientific World Journal*, vol. 2014, pp. 1–10, 2014.

[75] S. Trenn, "Multilayer perceptrons: Approximation order and necessary number of hidden units," *IEEE Transactions on Neural Networks*, vol. 19, no. 5, pp. 836–844, 2008.

[76] S. Keerthi and C.-J. Lin, "Asymptotic behaviors of support vector machines with gaussian kernel," *Neural Computation*, vol. 15, no. 7, pp. 1667–1689, 2003.

[77] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of the 13th International Conference on Machine Learning*. Morgan Kaufmann, 1996, pp. 148–156.

[78] G. Holmes, B. Pfahringer, R. Kirkby, E. Frank, and M. Hall, "Multiclass alternating decision trees," in *Proceedings of the 13th European Conference on Machine Learning*. Springer, 2002, pp. 161–172.

[79] 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 and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[80] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[81] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[82] T. Tommasi and B. Caputo, "The more you know, the less you learn: From knowledge transfer to one-shot learning of object categories," in *Proceedings of the British Machine Vision Conference*. British Machine Vision Association, 2009, pp. 1–11.

[83] K. R. Canini, M. M. Shashkov, and T. L. Griffiths, "Modeling transfer learning in human categorization with the hierarchical Dirichlet process," in *Proceedings of the 27th International Conference on Machine Learning*, J. Fürnkranz and T. Joachims, Eds. Omnipress, 2010, pp. 151–158.