Keypoints Detection and Feature Extraction: A Dynamic Genetic Programming Approach for Evolving Rotation-invariant Texture Image Descriptors

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

***Abstract*—The goodness of the features extracted from the instances and the number of training instances are two key components in machine learning, and building an effective model is largely affected by these two factors. Acquiring a large number of training instances is very expensive in some situations such as in the medical domain. Designing a good feature set, on the other hand, is very hard and often requires domain expertise. In computer vision, image descriptors have emerged to automate feature detection and extraction; however, domain-expert intervention is typically needed to develop these descriptors. The aim of this paper is to utilise Genetic Programming to automatically construct a rotation-invariant image descriptor by synthesising a set of formulae using simple arithmetic operators and first-order statistics, and determining the length of the feature vector simultaneously using only two instances per class. Using seven texture classification image datasets, the performance of the proposed method is evaluated and compared against eight domain-expert hand-crafted image descriptors. Quantitatively, the proposed method has significantly outperformed, or achieved comparable performance to, the competitor methods. Qualitatively, the analysis shows that the descriptors evolved by the proposed method can be interpreted.**

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

I. Introduction

THE process of developing an image classifier requires an appropriate set of features and a classification method. The term *feature* in this paper refers to a measurable property such as the number of pixels, or the average intensity value of a specific region of the image. Finding appropriate features may be more important than designing an effective classification algorithm since, in many cases, this requires a domain-expert [1], [2]. If the extracted features are informative, even a very simple classification model, e.g., k-Nearest Neighbour, can be sufficient to achieve good classification performance [3]. In order to construct a feature vector for an image, a set of *keypoints*, i.e., regions of interest, need to be identified first. An example of keypoint identification is using legs and wheels to discriminate between images where each is either a horse or a vehicle. In texture images, some typical keypoints are lines, corners, circles, and spots. Usually a domain-expert is required to define those informative keypoints. The second step is to detect those keypoints in an image, i.e., find their corresponding pixel coordinates. Conventionally, this task is performed manually where a domain-expert has to label the coordinates of each keypoint in every image, which is an expensive and time-consuming task. This has motivated researchers to automate this task and numerous algorithms have been proposed that aim at detecting a specific keypoint, e.g., corners [4]–[6], edges [7], and ridges [8], or a set of keypoints [9], [10]. The third step is to extract a value or a set of values from a detected keypoint such as calculating the average value of pixel intensities, contrast, homogeneity, or correlation. Automating the second and third steps, and combining them into a single model, is known as *image descriptor* in computer vision and pattern recognition [11]. Developing image descriptors has attracted many researchers and received increasing attention over the past few decades. Examples of commonly used image descriptors are Gray-level Co-occurrence Matrix (GLCM) [12], Local Binary Patterns (LBP) [9], Scale-invariant Feature Transform (SIFT) [13], Speeded-Up Robust Features (SURF) [14], KAZE Features [15], and Fast Retina Keypoint (FREAK) [16]. Based on how they operate, image descriptors can be at least categorised into *dense* and *sparse* [11], [17]. Dense descriptors operate in a pixel-by-pixel fashion such as LBP; whereas sparse descriptors such as SIFT consider only some pixels or parts of an image. Although the process of detecting those keypoints and extracting features have been automated, the intervention of a domain-expert is still required to perform the first step, i.e., keypoint identification. Furthermore, how to automatically detect those keypoints and what features can be extracted from the detected keypoints are determined by a domain-expert (e.g. GLCM and SURF). Another important issue is that some of these descriptors are not robust to image variants such as illumination, rotation, and scale. Conventional LBP is a typical example that has been extended in order to handle illumination and rotation.

The length of the feature vector extracted from an image is often predetermined and static. For example, conventional LBP produces a feature vector with length 2^p where p is the number

H. Al-Sahaf, M. Zhang, and A. Al-Sahaf 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@ecs.vuw.ac.nz; mengjie.zhang@ecs.vuw.ac.nz; ausama.alahaf@gmail.com).

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).

Algorithm 1 The GP Evolutionary Process

Input: $\mathcal{T}, \mathcal{F}, \delta, \beta, \gamma$ ▷ Terminal ($\mathcal{T}$) and function ($\mathcal{F}$) sets, population size (δ), generations (β), and ideal fitness (γ).
Output: ϑ ▷ Best evolved program
$i \leftarrow 0$ ▷ The i^{th} generation
$\lambda \leftarrow +\infty$ ▷ Best fitness so far
$\vartheta \leftarrow null$ ▷ Best solution so far
$\Xi_0 \leftarrow$ GENERATE$(\mathcal{T}, \mathcal{F}, \delta)$ ▷ The initial population
repeat
  for all $\xi \in \Xi_i$ **do**
    $\Delta_\xi \leftarrow$ FITNESS(ξ) ▷ Fitness of the current individual
    if $\Delta_\xi < \lambda$ **then** ▷ A better fitness than so far
      $\lambda \leftarrow \Delta_\xi$
      $\vartheta \leftarrow \xi$
    end if
  end for
  $\Xi_{i+1} \leftarrow$ POPULATE(Ξ_i) ▷ Populating subsequent generation
  $i \leftarrow i + 1$ ▷ Increment the generations counter
until $(i = \beta$ **or** $\lambda = \gamma)$ ▷ Check the termination criteria
return ϑ

of neighbouring pixels; whereas uniform LBP (LBP^{u2}) [18] generates a feature vector with length $p(p-1)+3$. Specifying the length of the feature vector increases the number of parameters required to be set. This task can be accomplished empirically; however, computationally it can be a very expensive task to perform [19].

Genetic Programming (GP) is a widely used Evolutionary Computation (EC) technique that evolves (searches for) a solution (a computer program), for a user defined problem via simulating the principles of natural selection and survival of the fittest [20], [21]. Generally, EC techniques start from a set/population of randomly generated candidate solutions that will be improved gradually over a number of cycles/generations. The process is guided using a fitness measure that reflects the goodness of each individual in the population to tackle the problem being solved. Algorithm 1 depicts the GP evolutionary process.

Since its introduction, GP has been used to tackle many image-related problems such as feature extraction [22], [23], classification [24], [25], object detection [26], [27], image segmentation [28], [29], image registration [30], and image processing [31]. Ebner and Zell [32] employed GP to automatically evolve an interest point detector. A similar approach was adopted by Trujillo and Olague [33] where GP is used to synthesise an interest point detector. The preliminary work of Trujillo and Olague was extended in [34] to improve the performance of the evolved interest point detector through considering the geometric stability and global separability of the detected points. GP is employed by Olague and Trujillo [35] to combine image operators, e.g., histogram normalisation and Gaussian smoothing, that aim at detecting interest points in an image. Following the same direction, Perez *et al.* [36] utilised GP to construct image descriptors for detecting objects in an image. Shao *et al.* [37] adopted a multi-objective GP approach to combine image processing operators, e.g., Laplacian, and Gabor filters, for image classification. Using GP to recognise human actions by evolving a spatio-temporal descriptor is the main focus of Liu *et al.* [38].

Many of these works employ different image processing techniques such as filtering, image derivatives, and convolution. Moreover, many are designed to identify or detect a specific type of keypoint, e.g., lines or corners. Non-EC developed image descriptors, on the other hand, are designed to detect a wide variety of keypoints. For example, LBP and its variants are typical examples that have the potential to detect lines, corners, spots, and edges. However, almost all LBP-based image descriptors require careful design of how to generate code at each pixel, and hence, a domain-expert is needed to design mathematical formulae to accomplish this task. Furthermore, changing one of the parameters, e.g., number of neighbouring pixels or radius, may require substantial changes to those formulae.

In machine learning, it is well known that a sufficiently large training dataset (in terms of the number of instances) is required in order to achieve a good level of performance [39]–[41]. However, acquiring a sufficiently large number of labelled instances, where each instance has the corresponding class label or labels if each instance comprises more than one object, is costly or sometimes infeasible [40], [42]. The medical domain is a typical example where often only a few labelled instances are available mainly because an expert is required to highlight the regions of interest, and such experts are often very expensive to employ. Another key factor is that using a large training data set imposes a heavy computational load on the learning system to evolve or train a model [40]. Zhu *et al.* [43] investigated the trade-off between the complexity and the size of the training data set on the model performance to perform object detection in images. They observed that increasing the training data can help, but the assumption is that the added data must have correct regularisation (system complexity) and treatment (preprocessing) of noisy instances. In other words, additional data may degrade the performance as it increases the possibility of noisy instances. In some of our recent work [19], [44]–[46], we also observed that GP can evolve a good keypoints detection or feature extraction model using only a limited number of training instances. Both [19][1] and [46] are directly related to the work of this study. Although the aim of the method proposed in [46] is to tackle the length of the feature vector via introducing special nodes ($expand$ and $switch$) in the program representation, the experiments are rather limited since only two datasets are used, and more importantly, those datasets are rotation-free. Furthermore, we have identified a major drawback of this method (more details in Section II-A). The method in [19], on the other hand, has successfully been designed to evolve rotation-invariant image descriptors and extensively examined; however, the length of the feature vector is static and a parameter tuning phase is needed to determine this parameter. The methods in [44], [45], and [46] were not able to cope with rotation as we have observed in our experiments. Hence, this paper substantially extends those methods to evolve rotation-invariant image descriptors, dynamically determining the length of the feature vector, extensively evaluating the evolved descriptors, and providing a deep analysis.

[1]The method proposed in [45] has been extended to handle rotation in [19].

A. Goals

This paper aims at using GP to automate the process of constructing a rotation-invariant image descriptor that detects a set of automatically designed keypoints, i.e., the user does not specify those keypoints (such as corners and edges), and extracts informative features from those keypoints; simultaneously, the system automatically determines the length of the feature vector. Motivated by the success of [19], a set of simple arithmetic operators and first-order statistics (e.g. mean and standard deviation) are automatically synthesised as a set of formulae that form an image descriptor, i.e., an evolved GP program. More importantly, this method does not require human intervention to design the keypoints and features; instead it uses a small sample (only two instances) of each class to evolve a descriptor. Moreover, the length of the feature vector is dynamic and will be determined during the evolutionary process. Quantitatively, the performance of an automatically constructed image descriptor by GP will be compared to that of eight domain-expert designed descriptors using ten commonly used machine learning classification methods on seven image benchmarks for multi-class texture classification. Those datasets are of varying difficulty, and are comprised of a different number of classes and rotations. Qualitatively, on the other hand, a constructed descriptor will be closely examined to shed light on how the proposed method can perform well. The following objectives will be investigated in this study.

- Develop a new tree-based [21] GP program representation that allows a node to have a dynamic number of children.
- Assess the evolved descriptors quantitatively and compare them to eight domain-expert designed descriptors on seven texture image benchmarks.
- Provide a qualitative assessment by investigating the interpretability and other aspects of an evolved descriptor.

Note that the work of this paper is a substantial extension to our recent works [46] and [19]. The method proposed in this paper is specifically designed to effectively handle the rotation variation and simultaneously determine the length of the feature vector. Here, the fitness measure, program representation (terminal and function sets), and experiment design are newly developed or substantially extended to overcome the limitations of the baseline methods ([46] and [19]). Furthermore, the method proposed in this study is evaluated both quantitatively and qualitatively.

B. Organisation

The remainder of the paper is organised as follows. The background and a survey of related literature are briefly discussed in Section II. Section III describes the proposed method. Section IV presents the experiment design. The results are presented and discussed in Section V. A descriptor evolved by the proposed method is thoroughly examined in Section VI. Section VII concludes this study and presents some directions for future research.

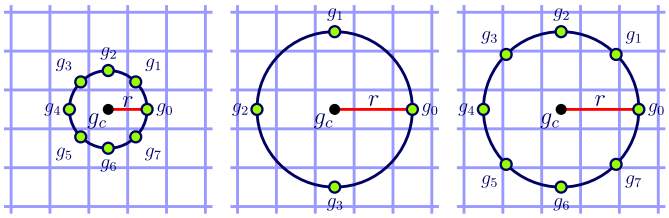


Fig. 1. Illustration of the $\text{LBP}_{p,r}$ parameters (a) $\text{LBP}_{8,1}$, (b) $\text{LBP}_{4,2}$, and (c) $\text{LBP}_{8,2}$.

II. Literature Survey

A brief background on the directly related work is provided in the first part of this section. The second part discusses some of the currently existing methods in the literature regarding GP for keypoints detection, feature extraction, and classification.

A. Background

The methods proposed in [19], [45], [46] as well as the method proposed in this study are largely designed to operate in a similar scheme to conventional LBP. Hence, LBP is briefly introduced first, followed by a discussion on the baseline methods GP-criptor [45], and EID [46].

1) Local Binary Pattern (LBP): A well-recognised and widely used image descriptor in computer vision is the *local binary pattern* (LBP) [9]. This image descriptor operates in a pixel-by-pixel manner, and aims at detecting a variety of image keypoints and generates a histogram (feature vector). Each bin in the histogram corresponds to the frequency of a specific keypoint [47]. The process of generating a feature vector for an image starts by initialising (set all counts/elements to zero) a histogram of length 2^p, where p is the number of neighbouring pixels. Then a binary code is generated at each pixel of the image by comparing the intensity of the current pixel and each of its circular equidistant neighbours. A parameter r specifies the distance, i.e., radius, and the effect of using different combinations is illustrated in Fig. 1. The binary code is then converted into a decimal value and the corresponding bin of the histogram is incremented by 1. This process is depicted in Fig. 1, and is defined formally as follows:

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

$$g_i = I(x_i, y_i) \tag{2}$$

$$x_i = x_c + r\cos\left(2\pi i/p\right) \tag{3}$$

$$y_i = y_c - r\sin\left(2\pi i/p\right) \tag{4}$$

where I is an image of size $M \times N$, (x_i, y_i) are the coordinates of the i^{th} pixel, the coordinates of the current pixel is denoted by (x_c, y_c), and the value/intensity of the current and i^{th} neighbouring pixels are, respectively, denoted as g_c and g_i.

Although LBP was originally designed to extract texture features, this method and its variants have been broadly employed in a wide variety of applications, such as face detection and recognition [48]–[50], object detection [51],

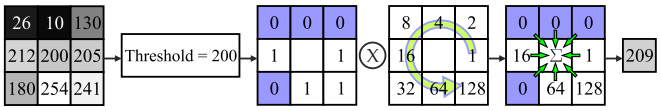


Fig. 2. Illustration of the LBP main steps.

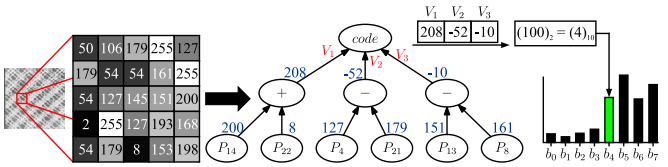


Fig. 3. Example presents a sample program evolved by GP-criptor and the main steps to generate the feature vector for an image.

[52], image segmentation [53], texture classification [54], and content based image retrieval [55]. Due to the simplicity of the algorithm and its effectiveness in detecting keypoints, the original proposal of LBP has been extended in different ways such as uniform LBP (LBP^{u2}) [18], uniform and rotation-invariant LBP (LBP^{riu2}) [56], median binary patterns (MBP) [57], soft/fuzzy LBP (FLBP) [58], [59], completed LBP (CLBP) [54], local ternary pattern (LTP) [60], local quinary patterns (LQP) [61], LBP with pyramid representation (PLBP) [62], local binary count (LBC) and completed LBC (CLBC) [63], robust LBP (RLBP) [64], and dominant rotated LBP (DRLBP) [65]. Reviewing LBP variants is beyond the scope of this study, and more details can be found in [50], [62], [63], [66], [67]. However, the proposed method in this study operates in a pixel-by-pixel manner (dense), and hence, it will be compared against common state-of-the-art dense descriptors such as LBP^{u2}, LBP^{riu2}, CLBP, LBC, CLBC, DRLBP, grey-level co-occurrence matrix (GLCM) [12], and domain-independent features (DIF) [26].

The two parameters r (radius) and p (number of considered neighbouring pixels) can potentially affect the design of LBP, specifically, the formulae to generate the binary code. Hence, human intervention is needed to modify or develop new formulae in order to have different settings for these two parameters; which in many cases can be a very difficult task. Furthermore, modifying conventional LBP to develop a descriptor that is robust to image variants, e.g., illumination and rotation, is not an easy task. The method proposed in this study aims at tackling these difficulties (rotation-invariant image descriptor, and the required formulae) by using GP to *automatically synthesise* a set of formulae (possibly non-linear) by combining simple arithmetic operators to form a rotation-invariant image descriptor. The newly introduced method does not require human intervention or background knowledge, and has a dynamic representation that allows the system to find an appropriate length of the generated code equivalent to the parameter p in LBP.

2) Genetic Programming Descriptor (GP-criptor): Motivated by the simplicity and effectiveness of LBP, and the flexibility and capability of GP to handle different data types, we have proposed a GP-based image descriptor (GP-criptor) [45]. GP-criptor aims at automatically constructing an LBP-like illumination-invariant image descriptor. Four simple arithmetic operators $\{+, -, \times, /\}$ along with a special $code$ node form the *function set*, and the pixel indices of a sliding window form the *terminal set*. In GP-criptor, only two instances of each class are randomly selected during the evolutionary process (training) to evolve the descriptor. The fitness function comprises of two components: *Accuracy*, and *Distance*. The former measures the ability of the features generated by the evolved descriptor to correctly classify the training instances using a simple instance-based classifier (k-Nearest Neighbour), whereas the *within-class* and *between-class* average distances are measured by the latter. Similar to LBP, an evolved program is used to generate the feature vector for an image by scanning the image pixel-by-pixel using a sliding window, generates a binary code, converts the generated code into a decimal value, and increments the corresponding bin of the histogram as presented in Fig. 3. At each position of the sliding window, the values of the terminal nodes are fed into the program, and those values are then passed to the parent nodes to evaluate each sub-tree/branch of the $code$ node children. The $code$ node uses the $s(\cdot)$ function in Equation (1) to specify the corresponding value of each child V_i, where $i \in \{1, 2, \ldots, q\}$ and q is the number of children. The generated binary code is then used to increment a bin in the histogram similar to conventional LBP.

The evaluation of GP-criptor in [45] is rather limited as only two datasets were used that are rotation-free. Evaluating GP-criptor on datasets with rotations revealed the limitation of this method to handle this variation [19].

3) Evolutionary Image Descriptor (EID): To tackle the problem of determining number of children of the $code$ node in GP-criptor, *Evolutionary Image Descriptor* (EID) was proposed [46]. EID has a lot in common with GP-criptor, such as the fitness function, terminal set, feature vector generating procedure, and the four arithmetic operators in the function set. However, two new functions were introduced in the function set of EID that allow the system to evolve programs with a dynamic number of bits in the binary code. The first node is $expand$, which does not perform any operation on its two children and only allows the system to grow by having chains of this type of node as presented in Fig. 4. The second node is $switch$, which has a single child and performs a threshold on the value of that child similar to the $s(\cdot)$ function in Equation (1). Each $switch$ node represents a bit in the binary code that is generated at each position of the sliding window. For example, the program in Fig. 4 consists of three $switch$ nodes, hence, the length of the generated feature vector for an image is 2^3 bins. An $expand$ node can only appear at the top part of the tree including the root, or as a child of another $expand$ node; whilst a $switch$ node can only be the child of an $expand$ node or the root (if the tree does not have any $expand$ nodes). Although the system became more powerful and reduced the number of parameters to be set, it introduces two potential problems. First, very large trees with an unnecessarily large number of $switch$ nodes means the feature vector will be excessively long, as has been observed in [46]. Second, the depth of each $switch$ node, i.e., the

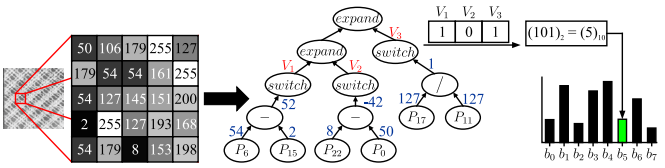


Fig. 4. Example presents a sample program evolved by EID and the main steps to generate the feature vector for an image.

sub-tree, will be very shallow as more $expand$ nodes are occupying a large proportion of the evolved program. Al-Sahaf *et al.* [46] proposed a way to overcome the former problem via identifying the indices of all those bins having 0 value across all training instances, and then remove those bins from all training, as well as the unseen, instances. The latter problem remains unsolved and an evolved program might not have sufficient combinations of operators for each $switch$ node, which will result in a lack of responses for different instances.

In addition to those newly introduced issues regarding the $expand$ and $switch$ nodes, similar to GP-criptor, EID cannot handle the rotation variant as it was not designed to cope with this problem. The proposed method in this paper, however, is specifically designed to overcome these limitations.

B. Related Work

EC techniques have been widely used to tackle many image-related problems [68]–[70]. However, the flexibility of GP representation, ability to handle different types of data, and interpretability, have attracted many researchers over the last few decades to consider this EC technique to tackle image-related problems in a wide variety of applications, among which are feature extraction, feature selection, keypoints detection, and classification. A brief discussion regarding some the most related work is provided in this subsection.

Song *et al.* [71] is one of the earliest works using GP to evolve a classifier that operates directly on the pixel values for texture image classification. By adopting two ideas, namely *Static Range Selection* (SRS) [72], [73] and *Dynamic Range Selection* (DRS) [74] to tackle the multi-class classification problem, Song *et al.* [71], [75] tested their methods for texture classification and the results of their works show very good performance compared to the other methods.

To tackle the limitations of SRS and DRS approaches, Smart and Zhang [76] developed two methods for multi-class classification tasks in GP: Centered Dynamic Range Selection (CDRS), and Slotted Dynamic Range Selection. They assessed their methods using five image datasets of increasing difficulty and promising performance was observed.

The multi-class classification problem is tackled in [77] by introducing a modified GP program representation called *modi* that is capable of generating multiple values rather than the single value generated from the root node in conventional GP. Testing the methodology for object detection using four image datasets showed the superiority of modi over conventional and other GP-based methods. However, modi is not designed to operate directly on raw pixel values and requires a set of pre-extracted features.

Training a good classifier using a highly unbalanced dataset (where there is a large difference between the number of training instances in each class) is a very challenging task. Bhowan *et al.* [78], [79] used GP to tackle this problem in a variety of ways for binary classification *without* under or over sampling.

Using Strongly-Typed GP (STGP) [80], Al-Sahaf *et al.* [81] proposed a multi-layer GP representation that comprises of two layers: aggregation, and classification; therefore, it is called *Two-tier GP* (2TGP). The aggregation layer extracts features from different regions of the image, whereas the classification layer assigns a class label to the image being evaluated. The different aspects and characteristics of 2TGP have been extensively studied for binary image classification tasks [22], [82], [83] and were shown to outperform competitive methods. Although 2TGP is an automated system that performs region of interest detection, feature extraction, and classification, a large number of training instances are required to evolve a good model.

Combining GP and SIFT features to improve the performance for object recognition is proposed by Hindmarsh *et al.* [84]. The idea is to use GP as a post-processing step to construct better features from the detected SIFT features. The results in [84] are comparable to the use of SIFT features alone. The system operates in two stages where keypoints detection and feature extraction are performed by SIFT in the first stage, and feature construction and classification are performed in the second stage by GP. However, abundant labelled data is needed.

Ryan *et al.* [85] used GP techniques for detecting stage-1 cancer in digital mammograms. Their method performs a series of preprocessing operations, e.g., background suppression, image segmentation, feature detection, and feature selection, in order to reduce the volume of data to process, and then the preprocessed data are fed into GP to evolve a classifier. The results of their experiments show that this workflow (preprocessing steps and GP classifier) can successfully detect a stage-1 cancer in digital mammograms. The workflow requires task-specific features which requires domain-expert intervention. As a multiple stage system, the success of any subsequent stage is subject to the goodness of performing previous stages.

Detecting edges in images is an important task in a wide variety of applications in computer vision such as image segmentation. Fu *et al.* [86], [87] studied edge detection and used GP to evolve edge detectors that outperformed some well-known detectors that are designed by domain-experts such as Sobel [88], [89] and Canny [7]. However, their methods are designed to detect only one type of keypoint, i.e., edges, while neglecting other types of keypoints, and requires a large number of training examples.

By adopting a multi-objective approach, Albukhanajer *et al.* [23] utilised GP for extraction of image features that are robust to noise and invariant to geometric deformations, e.g., illumination, rotation, and scale. Their system automatically combines different functionals and aims to maximise the between-class variance and minimise the within-class variance. The results of their experiments on two datasets showed good

robustness of the method to noise and geometric deformations.

Recently, we have proposed GP-criptor$^{\text{ri}}$ [19] (an extended version of GP-criptor) to address the limitation of GP-criptor to handle rotation. GP-criptor$^{\text{ri}}$ uses a set of first-order statistics to form the terminal set instead of the indices of the sliding window. GP-criptor$^{\text{ri}}$ has been extensively evaluated in [19] using six image datasets for texture classification that comprise a different number of classes, instances, and rotations; and compared against the performance of hand-crafted image descriptors. Although GP-criptor$^{\text{ri}}$ has been shown to significantly outperform, or achieve comparable performance to, the competitive hand-crafted image descriptors, this method requires performing a parameter tuning phase in order to set the number of children for the *code* node. The number of children of *code* plays a crucial role as it specifies the length of the feature vector. Empirically setting the number of children is a very time-consuming task as has been observed in [19].

In summary, most of the currently existing methods require abundant training examples in order to achieve a satisfactory level of performance as they were not designed to tackle the absence of enough labelled data. Those methods were designed to evolve a classifier/detector to handle image classification or object detection, rather than constructing an image descriptor. Some of those methods cannot operate directly on the raw pixel values; instead, pre-extracted features are required. Hence, domain-expert intervention is required to select/design a set of good keypoints and perform feature extraction. The existing image descriptors, on the other hand, were designed by domain-experts and have been successfully employed to tackle numerous problems in a wide variety of applications in computer vision. Altering those descriptors to handle different image deformations, or even a parameter, may require changing the system substantially. Another key factor of those descriptors is that the domain-expert might miss some of the good/crucial keypoints during the design phase and recovering those good keypoints will be infeasible, if not impossible, in the later phases. This paper aims at tackling those issues by utilising GP to evolve image descriptors using only two instances per class, which automatically detects keypoints and extracts features from those keypoints. Simultaneously, the proposed method in this study aims to dynamically set the length of the feature vector. The new method is described in detail in the next section.

III. The Proposed Method

The proposed *Rotation-invariant Evolutionary Image Descriptor* (EID$^{\text{ri}}$) method is described in this section. To make this paper self-contained and provide essential details, the inherited components from GP-criptor$^{\text{ri}}$ [19] are presented/discussed in this section. The overall algorithm is discussed first. Then the program representation is explained. Finally, this section describes the fitness measure and the procedure of extracting the feature vector from an image.

A. The Overall Algorithm

The overall process is depicted in Fig. 5. The instances of each class are equally split between the training and test sets. A key factor of the proposed method is that only a few instances of each class are required to evolve a descriptor. Therefore, the system randomly selects a subset (two instances per class) of the training set and feeds it into GP. The final result of the GP evolutionary process is an image descriptor that takes an image and produces a feature vector. The randomly selected training instances, i.e., those used during the evolutionary process, and the test set are then fed into the evolved descriptor to generate the transformed training set ($\mathbf{S_{tr}}$) and transformed test set ($\mathbf{S_{ts}}$) respectively. The $\mathbf{S_{tr}}$ is used to train a classifier which is then evaluated using $\mathbf{S_{ts}}$. Here $\alpha = \{(\vec{x}_i, c_i)\}\ \alpha \in \mathbf{S_{tr}} \cup \mathbf{S_{ts}}$ and $i \in \{1, 2, \ldots, z\}$, where $\vec{x}_i \in \mathbb{R}_{\geq 0} = \{l \in \mathbb{R} \mid l \geq 0\}$ and $c_i \in \{t_1, t_2, \ldots, t_C\}$ denote the i^{th} instance's feature vector and corresponding class label. The total number of classes is C, and the total number of instances is z.

B. GP Process

Using the given terminal and function sets, GP generates the initial population using the *ramped-half-and-half* method [21]. Fig. 6 presents an example of an EID$^{\text{ri}}$ individual. The fitness of each individual program is then measured on the training set (i.e. the two instances of each class). GP then generates a population of new programs in the subsequent generation by applying the different genetic operators (crossover, mutation and reproduction) on some individuals selected from the population in the current generation. Iteratively this process continues until the maximum number of generations is reached, and the best evolved program (image descriptor) is then returned. The program representation, fitness measures, feature vector extraction, and the required algorithms are presented and discussed in detail in the following subsections.

C. Program Representation

Similar to GP-criptor$^{\text{ri}}$, the terminal set in EID$^{\text{ri}}$ consists of the *min*, *max*, *mean* and *stdev* nodes as presented in Fig. 6. Each of these nodes performs a simple first order statistic on a set of values. The *min* and *max* nodes, respectively, return the minimum, i.e., $\min(\cdot)$, and maximum, i.e., $\max(\cdot)$, value of a vector. The *mean* and *stdev* nodes, on the other hand, calculate and return the average and standard deviation as shown in Equation (5) and Equation (6), respectively.

$$mean = \frac{1}{|\vec{x}|} \sum_{l \in \vec{x}} l \tag{5}$$

$$stdev = \sqrt{\frac{1}{|\vec{x}| - 1} \sum_{l \in \vec{x}} (l - mean)^2} \tag{6}$$

Here $\vec{x}$ is a vector of elements, and $|\cdot|$ returns the length of a vector, i.e., the number of elements. These functions are order-independent, i.e., do not consider the indices of the elements in the vector, which is an important property to tackle the rotation variants as demonstrated in Fig. 7.

In order to keep the individual structure simple, the function set in EID$^{\text{ri}}$ consists of five functions. Four of these functions are the arithmetic $+$, $-$, $\times$ and protected $/$ operators. The $/$ function will return 0 if the numerator is 0, which is very important to prevent the occurrence of a "division by zero"

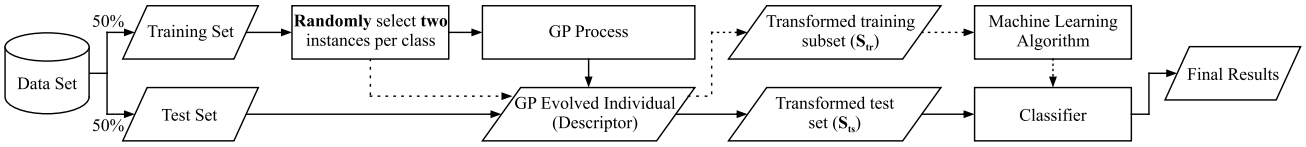


Fig. 5. Parts of the overall algorithm.

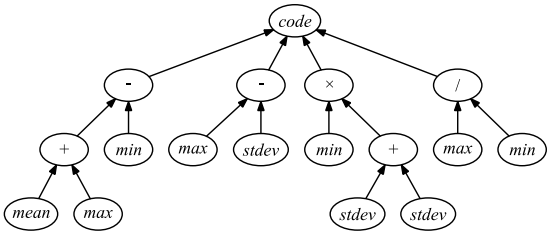


Fig. 6. Example of an evolved EID$^{\text{ri}}$ program.

Fig. 7. Example demonstrates the rotation-invariant property of the terminal nodes, (a) the original window, and (b) a 45° around the center rotated version of the same window.

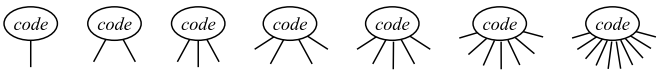


Fig. 8. Examples of *code* nodes with different number of children.

Algorithm 2 Measuring the fitness for an EID$^{\text{ri}}$ individual

function FITNESS($\mathbf{S}, r, ind, c, t$) ▷ Training images, radius, individual, number of classes, and number of instances per class
  $z \leftarrow (c \times t)$ ▷ Total number of instances in $\mathbf{S}$
  $\mathcal{A} \leftarrow \emptyset$ ▷ Empty set
  for i **in** $\{1, 2, \ldots, z\}$ **do**
    $\vec{f_i} \leftarrow$ FEATURES($\mathbf{S}_i, r, ind$) ▷ Algorithm 3 on i^{th} image
    $c_i \leftarrow$ CLASS($\mathbf{S}_i$) ▷ Class label of the i^{th} image
    $\mathcal{A} \leftarrow \mathcal{A} \cup \left\{ \left(\vec{f_i}, c_i \right) \right\}$ ▷ Concatenate the i^{th} image tuple
  end for
  $\{W_{\text{d}}, B_{\text{d}}\} \leftarrow$ DISTANCES($\mathcal{A}, c, t$) ▷ Algorithm 4
  return $1 \Big/ \left(1 + e^{-5(W_{\text{d}} - B_{\text{d}})}\right)$ ▷ Fitness of ind on $\mathbf{S}$
end function

exception. These functions are identical to that of GP-criptor$^{\text{ri}}$ where each performs the corresponding operator on two inputs and returns the resulting value. The main difference between GP-criptor$^{\text{ri}}$ and EID$^{\text{ri}}$ is the *code* function, which is the fifth component of the function set. This function is different from the other four functions in three ways. First, the number of inputs, i.e., children, is dynamic and is automatically determined by GP during the evolutionary process. However, the number of children must be greater than 0; otherwise, this node will not return any value. Second, this function thresholds the input values such that all negative values are substituted with 0 and all positive and zero values are substituted with 1; which makes the type of the input and output values different. Therefore, the *code* node can not appear anywhere apart from being the root node of a program tree. Third, each individual can have only a single *code* node due to the input-output type-mismatch. The system generates individuals that have different number of children under the *code* node. To accomplish this, the system is provided with a list of *code* nodes each of which has a specific number of children, e.g., ranging between 1 and 10, and the system randomly chooses one of these nodes to build an individual as depicted in Fig. 8.

D. Fitness Measure

Typically, the number of instances that are used during the evolving/training process is relatively large to address different variations of each class, and therefore, it is assumed that a well-trained model will give a satisfactory performance on the unseen data. In this case, measuring the accuracy, which is the proportion of those instances correctly classified to the total number of instances, represents an adequate measure. An important objective of this study is to evolve image descriptors using only a few training instances. When the number of training instances of each class is relatively small, e.g., less than 10, the use of accuracy often becomes insufficient, mainly because the system can easily capture features that are good enough to discriminate between the training instances, but not sufficient to classify the unseen data. This is an example of the well-known phenomenon in machine learning called *over-fitting* [90], [91]. Therefore, it is necessary to derive an alternative fitness measure that encourages the system towards identifying a good set of representative keypoints. The aim is to detect keypoints that have a different pattern between instances belonging to different classes, and meanwhile, ensure that instances belonging to the same class are following the same pattern. Measuring the distance between the feature vectors is an alternative approach [19] that we have also used in this study as presented in Algorithm 4. Measuring the fitness for an individual evolved by EID$^{\text{ri}}$ is presented in Algorithm 2.

A large number of distance measures have been proposed in the literature. One of the widely used measures is χ^2 which measures the distance between two normalised vectors [92] as presented in Algorithm 5. The two vectors must be normalised and have the same number of elements. Formally, the χ^2 distance is:

$$\chi^2\left(\vec{u}, \vec{v}\right) = \frac{1}{2} \sum_{i=1}^{E} \frac{\left(\vec{u}_i - \vec{v}_i\right)^2}{\left(\vec{u}_i + \vec{v}_i\right)} \tag{7}$$

where $\vec{u}$ and $\vec{v}$ are vectors, E is the number of elements in a vector, and $\vec{x}_i$ is the i^{th} element of $\vec{x}$.

Algorithm 3 Extracting the feature vector

function FEATURES($\mathbf{S}_i, r, ind$) ▷ Image, radius, and individual
  $q \leftarrow |code.children|$ ▷ Number of the children under $code$
  $\vec{f} \leftarrow (0_1, 0_2, \ldots, 0_{2^q})$ ▷ Set of length 2^q of zeros
  for each pix **in** $image$ **do**
    $\vec{x} \leftarrow$ PIXELS(pix, r) ▷ Neighbouring pixels
    $min \leftarrow$ MIN($\vec{x}$)
    $max \leftarrow$ MAX($\vec{x}$)
    $mean \leftarrow$ MEAN($\vec{x}$)
    $stdev \leftarrow$ STDV($\vec{x}$)
    $\mathbf{L} \leftarrow \{min, max, mean, stdev\}$
    $i \leftarrow$ EVALUATE($ind, \mathbf{L}$) ▷ Evaluate the tree on the inputs
    $\vec{f}_{\{i\}} \leftarrow \vec{f}_{\{i\}} + 1$ ▷ Increment the i^{th} element in $\vec{f}$
  end for
  return $\vec{f}$ ▷ Feature vector
end function

Algorithm 4 The between-class and within-class distances

function DISTANCES($\bar{\mathbf{S}}, c, t$) ▷ Set of tuples $\{(\vec{x}_i, c_i)\}$ where $i \in \{1, 2, \ldots, |\bar{\mathbf{S}}|\}$, number of classes, and number of instances per class
  $z \leftarrow (c \times t)$ ▷ Number of instances in $\bar{\mathbf{S}}$, i.e., $|\bar{\mathbf{S}}|$
  for each $(\vec{x}_i, c_i)$ **in** $\bar{\mathbf{S}}$, $i \in \{1, 2, \ldots, z\}$ **do**
    for each $(\vec{x}_j, c_j)$ **in** $\bar{\mathbf{S}}$, $j \in \{1, 2, \ldots, z\} \setminus \{i\}$ **do**
      $distance \leftarrow$ DIST($\vec{x}_i, \vec{x}_j$) ▷ Algorithm 5
      if $c_i \neq c_j$ **then** ▷ Different classes
        $B_\mathrm{d} \leftarrow B_\mathrm{d} + distance$ ▷ Between-class sum
      else
        $W_\mathrm{d} \leftarrow W_\mathrm{d} + distance$ ▷ Within-class sum
      end if
    end for
  end for
  $B_\mathrm{d} \leftarrow B_\mathrm{d} / (z \times (z - t))$ ▷ Average between-class distances
  $W_\mathrm{d} \leftarrow W_\mathrm{d} / (z \times (t - 1))$ ▷ Average within-class distances
  return $\{W_\mathrm{d}, B_\mathrm{d}\}$ ▷ Between- and within-class distances
end function

Inherited from GP-criptor$^{\mathrm{ri}}$, this fitness function considers the *within-class* and *between-class* distances. Considering only the within-class distance, the system may evolve a program that can generate nearly similar feature vectors for instances belonging to different groups. In other words, the system will form only a single group/cluster as the aim is to minimise the distance between the instances. Meanwhile, the system may evolve a program that separates the instances such that each instance can form a cluster if the between-class distance considered alone. Therefore, the aim is to find a trade-off between these two distances by minimising the average distance between instances belonging to the same class and maximising the average distance between instances belonging to different classes as shown in Algorithm 4. In this way, GP will try to form a group/cluster for each class that keeps its instances close to each other, and simultaneously, makes those clusters separated apart as much as possible. The aim is that an unseen instance will be put into a cluster consisting of instances from the same class.

E. Feature Vector Extraction

The process of extracting the feature vector from an image in EID$^{\mathrm{ri}}$ is identical to that in GP-criptor$^{\mathrm{ri}}$. However, the length of the feature vector in GP-criptor$^{\mathrm{ri}}$ is predetermined by the user; whereas in EID$^{\mathrm{ri}}$ it is automatically determined by the system during the evolutionary process. The number of children (q) of the $code$ (root) node is used to initiate an empty histogram, i.e., feature vector, of length 2^q bins. This histogram is populated using a sliding window of a predetermined size (r) that scans the image being evaluated row-wise from the left-top corner to the right-bottom corner. At each pixel, i.e., position of the sliding window, the following operations are performed. The inputs (terminals) of the program's tree are calculated, e.g., minimum, maximum, mean, and standard deviation of the pixel values in the window. As the terminals become available, the program tree can be evaluated starting from the leaves up to the root (the standard GP procedure to evaluate an individual). At this point, the root node thresholds the values of its children in order to generate a binary code. All negative values map to 0; whereas zero and positive values map to 1. Similar to LBP, the generated code is then converted into the corresponding decimal value, and the bin at the index of this decimal value is incremented. This procedure is depicted in Algorithm 3.

Algorithm 5 Measure the distance between two vectors

function DIST($\vec{u}, \vec{v}$) ▷ Two vectors
  $E \leftarrow |\vec{u}|$ ▷ Number of elements ($|\vec{u}| = |\vec{v}|$)
  $ds \leftarrow 0$
  for i **in** $\{1, 2, \ldots, E\}$ **do**
    if $(\vec{u}_i + \vec{v}_i) \neq 0$ **then** ▷ Different classes
      $ds \leftarrow ds + \left((\vec{u}_i - \vec{v}_i)^2 \,/\, (\vec{u}_i + \vec{v}_i)\right)$
    else ▷ To prevent the division by zero exception
      $ds \leftarrow ds + 0$ ▷ Based on Cha [92]
    end if
  end for
  $distance \leftarrow (0.5 \times ds)$
  return $distance$ ▷ Distance between $\vec{u}$ and $\vec{v}$
end function

IV. Experiment Design

The performance of the proposed method is assessed by conducting a number of experiments using seven texture image datasets, and compared to well-known image descriptors in computer vision. The details of these datasets, parameter settings, methods for comparison, and implementation are provided in this section.

A. Data Sets

In this study, image classification is considered to evaluate the performance of EID$^{\mathrm{ri}}$. Image classification is concerned with assigning a class label to each instance. Seven multi-class image classification datasets are used in this study. Each consists of grey-scale texture images, and comprises a different number of classes. Moreover, the instances of these datasets differ in dimensions, number of instances, rotation angles, and illuminations.

1) Kylberg Texture: The *Kylberg Texture*[2] [93] dataset is widely used in computer vision for texture classification. It contains images of different materials, e.g., cushion, rug, rice,

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

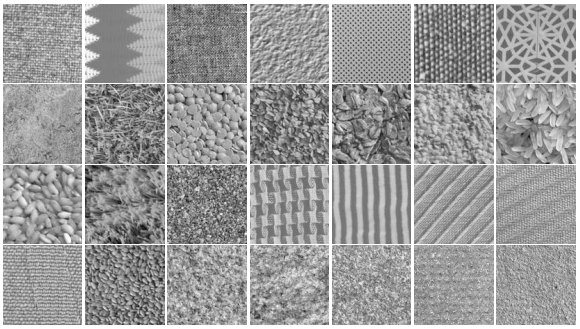


Fig. 9. Samples of the Kylberg dataset.

grass, and stone (Fig. 9). In total, this dataset comprises 28 classes each of which comes in two flavours: *without-* and *with-rotation*. This is very important as investigating the robustness of EID^{ri} to evolve rotation-invariant descriptors is a core aim of this study. Originally, the instances of this dataset were of size 576×576 pixels each and we have reduced the size to 115×115 pixels via subsampling. Each of the without-rotation classes consists of 160 instances while there are 1920 instances in each class of the with-rotation classes that fall into 12 rotation angles $\{0°, 30°, \ldots, 330°\}$. The instances of the without-rotation classes are used to form the *first* dataset in this study (KyNoRo), whereas the *second* dataset in this study (KyWiRo) is formed using the instances of the with-rotation classes.

2) Brodatz Texture: Another popular and widely used dataset for texture classification in computer vision is the *Brodatz Texture*[3] [94] dataset. Similar to Kylberg, the Brodatz dataset contains images for different materials, e.g., grass, bark, wood grain, and brick wall (Fig. 10). In total, there are 112 classes in the Brodatz dataset each of which consists of a single grey-scale instance of size 640×640 pixels. In order to generate the *third* dataset in this study (BrNoRo), the single instances of 20 classes are randomly selected then each instance is re-sampled into non-overlapping subimages each of size 84×84 pixels. Meanwhile, the same examples of those 20 classes are then rotated around the center by 12 angles with a step of size 30° to generate the *fourth* dataset in this study (BrWiRo). Hence, the total number of instances in BrNoRo is and BrWiRo is, respectively, 1680 ($= 20 \times 84$) and 20160 ($= 20$ (classes) $\times$ 12 (rotations) $\times$ 84 (instances)) instances.

3) Outex Texture Classification: The *fifth* (OutexTC00) and *sixth* (OutexTC10) datasets in our experiments are formed using two out of 16 datasets of the *Outex Texture Classification*[4] [95] dataset. Those 16 datasets comprise different numbers of classes, materials, rotations, and instances. The Outex_TC_00000 dataset is used to form OutexTC00, which comprises 24 classes each of which consists of 20 instances (Fig. 11). Meanwhile, OutexTC10 is formed using the instances of the Outex_TC_00010 dataset that also comprises of 24 classes each of which has 180 instances. The instances of OutexTC00 are rotation-free, whereas OutexTC10 instances

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

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

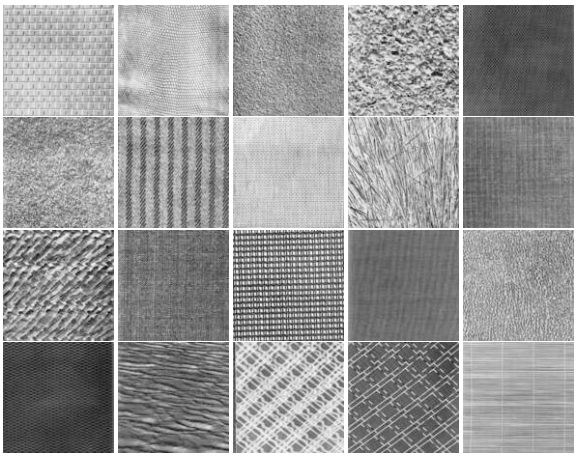


Fig. 10. Samples of the Brodatz dataset.

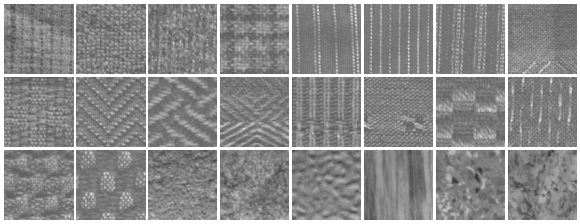


Fig. 11. Samples of the Outex Texture Classification dataset.

fall into 9 different angles: 0°, 5°, 10°, 15°, 30°, 45°, 60°, 75°, and 90°. Moreover, the instances in these *two* datasets are of size 128×128 pixels.

4) Kylberg Sintorn Rotation: The *seventh* dataset in this study (KySinHw) is formed using the *Kylberg Sintorn Rotation*[5] [96] dataset. Originally, this dataset consists of 25 texture classes, each made up of grey-scale instances of size 122×122 pixels as presented in Fig. 12. Each class comprises of 900 instances that fall into 9 different angles: 0°, 40°, 80°, 120°, 160°, 200°, 240°, 280°, and 320°. Moreover, the instances of this dataset are normalised with a mean value of 127 and a standard deviation of 40, and rotated using 6 different methods: *hardware*, *nearest neighbour*, *linear interpolation*, 3^{rd} *order cubic interpolation*, *B-spline interpolation*, and *Lanczos 3 interpolation*. In this study, only the instances that were rotated by the hardware method, where a sample is placed on a rotatable desk and the camera is positioned vertically on the top, are considered.

A summary of number of classes, instances, rotations, and dimensions for these datasets is presented in Table I.

B. Methods for Comparison

Since an image descriptor is only responsible for detecting keypoints and extracting features from those keypoints, the goodness of the extracted features to perform classification or detection is used in the literature as a measure to assess the performance of the descriptor [9], [13], [14], [48]. Therefore, different machine learning algorithms, Support Vector Machines (SVM), Naïve Bayes (NB), Adaptive

[5]Available at: http://www.cb.uu.se/~gustaf/KylbergSintornRotation/

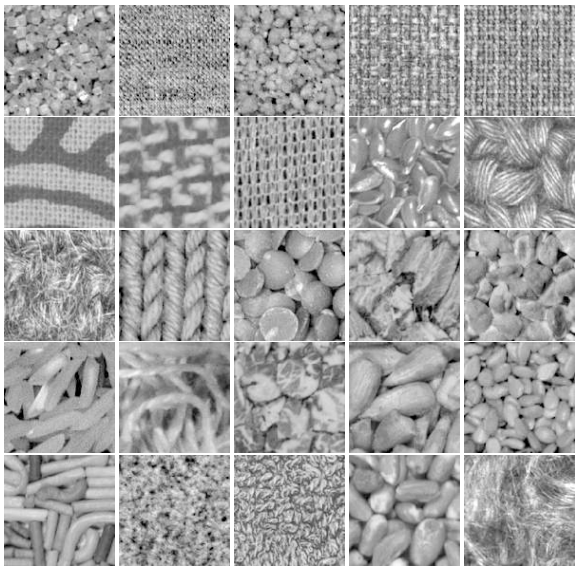


Fig. 12. Samples of the Kylberg Sintorn Rotation texture dataset.

TABLE I
A SUMMARY OF THE DATASETS.

Data set	Classes	Instances	Rotations	Dimensions	
				width	height
BrNoRo	20	1680	1	64	64
BrWiRo	20	20160	12	64	64
OutexTC00	24	480	1	128	128
OutexTC10	24	4320	9	128	128
KySinHw	25	22500	9	122	122
KyNoRo	28	4480	1	115	115
KyWiRo	28	53760	12	115	115

Boosting (AdaBoost), Decision Trees (J48), Random Forest (RF), Naïve Bayes/Decision Tree (NBTree), KStar (K*), Non-Nested generalised (NNge), k-Nearest Neighbour (k-NN), and Multilayer Perceptron (MLP), are used in this study to evaluate the performance of EID$^{\text{ri}}$. These classifiers are discussed in [97]. The effectiveness of EID$^{\text{ri}}$ is also investigated in this study by comparing its performance to a number of common state-of-the-art image descriptors such as DIF, GLCM, $\text{LBP}^{u2}_{p,r}$, $\text{LBP}^{riu2}_{p,r}$, $\text{CLBP}_{p,r}$, $\text{LBC}_{p,r}$, and $\text{CLBC}_{p,r}$.

C. Experiments

Two sets of experiments are designed and conducted in this study that aim at investigating different criteria. Apart from the evolutionary parameters, the proposed method has only one parameter that requires manual setting, that is the sliding window size. Therefore, the impact of using 3 window sizes, i.e., 3×3, 5×5, and 7×7, on the performance of a simple instance-based classifier (k-NN) using the features of an evolved image descriptor is investigated in the first experiment. This experiment can be seen as a parameter tuning stage. Investigating the influence of the evolved descriptors by EID$^{\text{ri}}$ on the performance of different types of classification methods and compared to other image descriptors (baseline methods) is the aim of the second set of experiments.

TABLE II
THE GP PARAMETERS.

Parameter	Value	Parameter	Value
Generations	50	Crossover Rate	80%
Population Size	300	Mutation Rate	20%
Minimum Depth	2	Maximum Depth	10
Selection Type	Tournament	Reproduction	Keep the best
Tournament size	5	Initial Population	Half-and-half

Similar to other EC methods, GP is a stochastic search method initialised with a random *seed* value. Therefore, for each experiment the proposed method has been independently executed 30 times using a different seed value each time and the best evolved program at the end of each run is reported. Moreover, the proposed method randomly selects two instances from each class to evolve an image descriptor. Using different instances for training could affect the evolved program. Hence, the process of 30 independent runs was further repeated 10 more times using different training instances each time. Then the average performance of those 300 ($=$ 30 (runs) $\times$ 10 (repetitions)) best individuals along with the standard deviation are reported. Saying that, the total number of runs for the first experiment is 7 (datasets) $\times$ 3 (windows sizes) $\times$ 10 (repetitions) $\times$ 30 (runs) $=$ 6300. Meanwhile, there are 7 baseline methods, 1 new method (using only the best window size found in the first experiment with 30 independent runs), 8 deterministic (single run each) and 2 stochastic (30 runs each) classifiers, 10 repetitions, and 7 datasets; therefore, there are 176120 experiments/runs in total in the second set of experiments.

D. Parameter Settings

This section discusses the parameter settings for the proposed method, methods for comparison, and the different classification algorithms.

1) Parameter settings for EID$^{\text{ri}}$: Since EID$^{\text{ri}}$ is a GP-based method, the GP evolutionary parameters are required. Due to the high computation costs of dealing with images, the population size is restricted to 300 individuals. The evolutionary process will be terminated when either the maximum number of 50 generations is reached, or an ideal solution, i.e., the fitness value is 0, is found. The minimum tree depth is 2 and the maximum depth is 10, which allow the system to evolve trees of different sizes but not too big which can slow down the evolutionary process and affect the interpretability of those trees. The mutation and crossover rates are, respectively, 20% and 80% which allow the system to focus on trying different combinations (crossover) before introducing a new materials (mutation) [98], [99]. To ensure that the subsequent generation is at least as good as the current generation, the best individual of the current generation is copied without changing (i.e. elitism) to the next generation. The initial generation is generated by the commonly used ramped-half-and-half method. The tournament method is utilised for selecting individuals for the mating process. These parameters are summarised in Table II.

2) Parameters of the baseline methods: As depicted in Fig. 13, we have observed that the radius (r) and the number

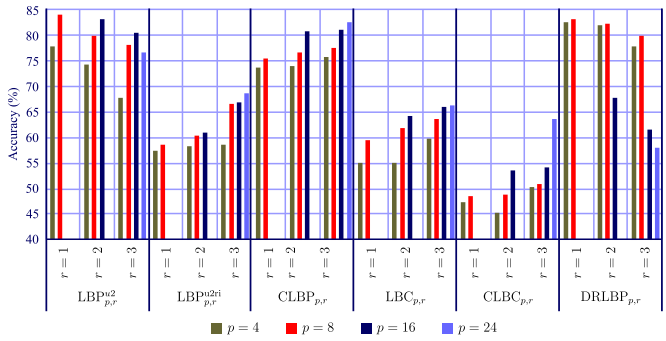


Fig. 13. The impact of p and r on the performance of the LBP-based image descriptors on the BrNoRo dataset.

of neighbouring pixels (p) for $\text{LBP}^{\text{u2ri}}_{p,r}$, $\text{CLBP}_{p,r}$, $\text{LBC}_{p,r}$ and $\text{CLBC}_{p,r}$ have achieved the best performance when they are, respectively, set to 3 and 24. Meanwhile, these two parameters, i.e., r and p, have been set to 1 and 8 for $\text{LBP}^{\text{u2}}_{p,r}$ and $\text{DRLBP}_{p,r}$ as the experiments revealed that this combination shows the best performance compared with the other settings.

3) Parameters of the classifiers: In this study, a number of well-known classification methods are used to assess the goodness of the generated features by an image descriptor evolved by EID^{ri} or one of the baseline methods. Those classifiers, as discussed in Section IV-B, have several parameters and tuning all of them is beyond the scope of this study. However, some of the important ones are considered in our experiments. Due to having only two instances of each class in the training set, the number of the neighbours for all instance-based methods, e.g., NNge, k-NN, and K*, is set to 1 (the closest neighbour).

For SVM, a study by Keerthi and Lin [100] investigated the impact of linear and non-linear kernels on the performance. In their study, it has been observed that using non-linear kernels in SVM are more likely to give better performance than using linear kernels. Following their suggestions, the *Radial Basis Function* (RBF) kernel is employed in our experiments.

Similarly, the network structure of MLP is specified based on the guidelines of Trenn [101]. Typically, MLP has a single input layer that comprises of one node for each feature in the feature vector; whereas the number of classes specifies the number of nodes in the output layer. Only one hidden layer is used and the following formula is considered to calculate the number of nodes in this layer:

$$N_{\text{hidden}} = \left\lceil \frac{N_{\text{in}} + N_{\text{out}}}{2} \right\rceil \tag{8}$$

where the values of N_{in} and N_{out} correspond to the number of nodes in the input and output layers.

AdaBoost is a meta-algorithm that was introduced by Freund *et al.* in 1996 [102] designed to be used in conjunction with other machine learning algorithms in order to enhance their performance. The overall idea of AdaBoost is to adaptively build a model by considering those previously misclassified instances by the current models to improve the subsequent ones. In our experiments, the *muLti-class Alternating Decision Trees* (LADTree) classifier [103] is used as it gave better performance than *DecisionStump*.

E. Implementation

The proposed method is implemented using the platform provided by the Evolutionary Computation Java-based (ECJ) package version 23 [104]. The Waikato Environment for Knowledge Analysis (WEKA) package version 3.8 [105] implementations for all other aforementioned classifiers (Section IV-B) are used.

V. Results and Discussions

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

A. Window Size

The aim of the first experiment is to investigate the impact of using different window sizes on the performance of the proposed method, and the obtained results are presented in Fig. 14. On the BrNoRo and BrWiRo datasets, the observed performances for different window sizes were 90.21% and 92.12% for a 3×3 pixels window, 91.04% and 92.59% for a 5×5 pixels window, and 89.17% and 90.78% for a 7×7 pixels window, respectively. The results obtained on the KyNoRo and KyWiRo datasets in this experiment were 86.27% and 87.67% for a 3×3 pixels window, 86.90% and 88.60% for a 5×5 pixels window, and 85.78% and 86.51% for a 7×7 pixels window, respectively. On OutexTC00 and OutexTC10, the proposed method has achieved 87.51% and 85.90% for a 3×3 pixels window, 87.90% and 87.09% for a 5×5 pixels window, and 86.94% and 86.37% for a 7×7 pixels window, respectively. Finally, EID^{ri} obtained 92.85%, 94.06%, and 93.85% average accuracy using a window of size 3×3, 5×5, and 7×7 pixels, respectively. Clearly, the results of these datasets are following a similar pattern and the differences between using different window sizes is not large.

In summary, EID^{ri} has achieved better results with a window of size 5×5 pixels than those of the other two experimented window sizes, i.e., 3×3 and 7×7 pixels, for all datasets as depicted in Fig. 14. We did not investigate the use of larger window sizes such as 9×9 and 11×11 mainly because the results of the 7×7 window are generally worse than 5×5 window. Based on the results of this experiment, the window size has been set to 5×5 pixels in the second experiment.

B. Image Classification

The aim of the second set of experiments is to compare the effectiveness of EID^{ri} descriptors against a variety of LBP based and non-LBP based hand-crafted image descriptors that were developed by domain-experts. The results for all datasets are combined into a single table as presented in Table III. The first column of Table III lists the dataset name, while the name of the feature extraction methods, i.e., image descriptor, are listed in the second column. The performances of the different classifiers are listed in columns 3 to 12 each of which reports the average accuracy and standard deviation ($\bar{x} \pm s$). In order to correctly assess the significance of the results, it is very important to use a suitable statistical test. Testing the normality and

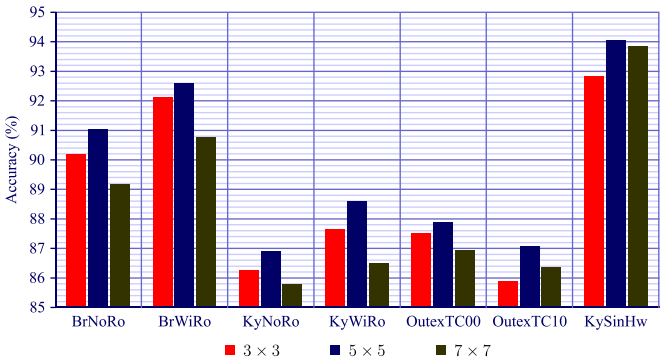


Fig. 14. The impact of the window size on the performance of EID^{ri} on the seven datasets.

homoscedasticity are required prior to the use of a parametric statistical test such as t-test and *analysis of variance* (ANOVA) [106], [107]. The results of the normality test revealed the skewness of the data, i.e., obtained results, which means the normality assumption is not accomplished. Therefore, the significance of the obtained results are tested using a non-parametric statistical test that is the *Wilcoxon signed-rank* test [106], [108] with a significance level of 5%. The statistical significance test has been performed to investigate how EID^{ri} with a simple 1-NN classification method competes with the other image descriptors using 1-NN and more powerful classifiers. The symbol "$*$" appears next to the method that has been significantly outperformed by EID^{ri}, and a "$-$" is used to indicate that the corresponding method has significantly better performance than that of EID^{ri}. Moreover, the statistical significance test has been applied again to assess whether the proposed method can compete with the baseline methods using the same classifier. In this case, the symbols "$\uparrow$" and "$\downarrow$" are used to, respectively, indicate that EID^{ri} is significantly better and significantly worse than the corresponding method. The overall best performance on each dataset is underlined, and the method with the best performance for each classification method is made **bold**.

1) BrNoRo: The results of the BrNoRo dataset are presented in the first block of Table III. The proposed method has achieved the overall best performance using 1-NN classification method with an average accuracy of 90.0%. The results of the first significance test show that EID^{ri} has significantly outperformed all the baseline methods even with more sophisticated classifiers. The second significance test revealed that EID^{ri} has significantly better performance than all the 8 baseline descriptors in 5 classifiers (excluding 1-NN), and at least better than 5 descriptors on the other 4 classification methods.

2) BrWiRo: On the BrWiRo dataset, i.e., the rotated version of BrNoRo, the results in the second block of Table III show that EID^{ri} has achieved the overall best average performance that is 92.6% on this dataset. Similarly, the proposed method with a simple instance-based classifier (1-NN) has significantly outperformed the other baseline hand-crafted descriptors on all 10 classifiers. Apart from $\text{CLBP}_{24,3}$ with MLP and SVM, EID^{ri} with the same classification method has achieved significantly better performance than all the 8 baseline descriptors as shown by the results of the second significance test.

3) KyNoRo: The third block of Table III shows the results obtained on the KyNoRo dataset. The newly introduced method with 1-NN shows the 4^{th} overall best performance as $\text{CLBP}_{24,3}$ achieved the first (MLP), second (1-NN), and third (NNge). Using 1-NN, EID^{ri} has significantly better performance than the other descriptors in the vast majority cases, and shows a significant improvement in the performance of 76.25% $(61/80)$ of the 10 classifiers compared to the use of those domain-expert designed descriptors. On the other 23.75% $(19/80)$ of the cases, the proposed method has comparable performance to the domain-expert methods for 11 out of 19 cases, and significantly degraded the performance on the other 8 cases.

4) KyWiRo: On the KyWiRo dataset, the proposed method shows nearly a similar pattern to that on the rotation-free version of this dataset (KyNoRo) as presented in the fourth block of Table III. Using the simple 1-NN classification method, EID^{ri} scored 88.6% accuracy on average which represents the fourth overall best performance on this dataset (KyWiRo) and has significantly outperformed the 8 baseline methods in over 88% of the cases for the 10 classification methods. The results of the second statistical test show that EID^{ri} has a significantly positive influence on the performance of the 10 classifiers compared to the use of the other image descriptors' features.

5) OutexTC00: As shown in the fifth block of Table III, the proposed method is the overall best performing method on the OutexTC00 dataset with 87.9% average accuracy. Apart from $\text{LBP}^{u2}_{8,1}$ with 1-NN, K*, MLP, and NNge, results of the first statistical test show that EID^{ri} with 1-NN has significantly outperformed all other methods with 1-NN and more powerful classifier. Using the same classification method, on the other hand, EID^{ri} has also shown a significant positive impact on the performance in the vast majority (over 87%) of those classifiers. However, in six cases the improvement was not significant and on other four EID^{ri} has either slightly or significantly degraded the performance.

6) OutexTC10: On the rotated version of the OutexTC00, i.e., OutexTC10, the proposed method shows the overall best performance that is 87.1% average accuracy as presented in the sixth block of Table III. The statistical results of using EID^{ri} with a 1-NN classifier against the baseline methods with 1-NN and other classifiers reveal that EID^{ri} has significantly better performance than the competitor methods apart from $\text{CLBP}_{24,3}$ with 1-NN, MLP, and NNge classifiers. The results of the second significance test show that the features extracted by the proposed method have significantly improved 67 out of 72 of the cases, and only slightly improved or degraded the performance of 5 cases. Only one case, i.e., $\text{CLBP}_{24,3}$ with MLP, shows significantly negative impact of EID^{ri} features.

7) KySinHw: The last (seventh) block of Table III lists the results obtained on the KySinHw dataset. The winner of the overall best performance on this dataset was $\text{CLBP}_{24,3}$ with NNge classification method, which achieved 97.9% accuracy on average. However, EID^{ri} is ranked fourth best performance (94.1%) amongst the other methods on the KySinHw dataset.

TABLE III
THE AVERAGE ACCURACY (%) OF TEN CLASSIFIERS USING NINE IMAGE DESCRIPTORS ON THE SEVEN TEXTURE IMAGES DATASETS ($\bar{x} \pm s$).

		1-NN	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∗↑
	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∗↑
	$LBP^{u2}_{8,1}$	84.0 ± 2.0∗	40.4 ± 4.7∗↑	33.6 ± 5.1∗↑	**86.2 ± 2.4**∗	83.4 ± 1.6∗	62.8 ± 8.3∗↑	71.4 ± 5.8∗↑	82.1 ± 2.7∗↑	56.5 ± 1.6∗↑	**75.3 ± 5.3**∗
	$LBP^{u2ri}_{24,3}$	68.5 ± 3.2∗	25.1 ± 4.3∗↑	39.0 ± 5.0∗↑	60.8 ± 3.5∗↑	68.8 ± 2.9∗↑	34.7 ± 7.2∗↑	46.7 ± 2.6∗↑	63.3 ± 3.9∗↑	48.4 ± 2.1∗↑	62.8 ± 3.6∗↑
	$CLBP_{24,3}$	82.4 ± 4.9∗	37.3 ± 8.0∗↑	35.3 ± 4.4∗↑	81.2 ± 4.5∗↑	**85.1 ± 3.5**∗	71.5 ± 5.0∗↑	77.4 ± 5.3∗↑	83.7 ± 4.7∗	61.3 ± 1.8∗↑	70.9 ± 5.7∗
	$LBC_{24,3}$	66.3 ± 2.8∗	23.5 ± 8.0∗↑	36.0 ± 6.8∗↑	58.2 ± 3.3∗↑	67.5 ± 3.1∗↑	30.2 ± 7.3∗↑	45.0 ± 3.1∗↑	61.7 ± 2.8∗↑	45.3 ± 2.3∗↑	62.9 ± 2.8∗↑
	$CLBC_{24,3}$	63.6 ± 2.5∗	32.5 ± 3.9∗↑	32.7 ± 3.9∗↑	67.1 ± 2.9∗↑	68.0 ± 2.2∗↑	61.9 ± 4.0∗↑	67.8 ± 5.9∗↑	65.9 ± 1.1∗↑	54.5 ± 1.4∗↑	50.7 ± 4.3∗↑
	$DRLBP_{8,1}$	83.2 ± 2.5∗	39.0 ± 4.7∗↑	35.7 ± 4.3∗↑	80.3 ± 3.2∗↑	83.6 ± 2.7∗	58.0 ± 10.1∗↑	71.0 ± 4.9∗↑	82.9 ± 3.8∗	59.7 ± 1.5∗↑	73.3 ± 6.4∗
	EID^{ri}	**91.0 ± 2.0**	**47.2 ± 2.6**	**51.8 ± 1.3**	85.4 ± 1.6	83.1 ± 1.7	**80.3 ± 2.7**	**82.2 ± 1.5**	**85.7 ± 1.9**	**70.3 ± 1.5**	71.8 ± 2.6
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∗↑
	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∗↑
	$LBP^{u2}_{8,1}$	42.3 ± 1.7∗	18.8 ± 2.7∗↑	20.1 ± 3.0∗↑	41.8 ± 2.0∗↑	40.9 ± 2.5∗↑	26.1 ± 3.6∗↑	33.0 ± 3.2∗↑	38.3 ± 3.3∗↑	26.3 ± 1.2∗↑	36.3 ± 4.0∗↑
	$LBP^{u2ri}_{24,3}$	67.6 ± 2.6∗	22.9 ± 6.4∗↑	40.1 ± 5.3∗↑	62.5 ± 2.4∗↑	68.6 ± 1.7∗↑	39.4 ± 4.5∗↑	49.1 ± 4.2∗↑	65.8 ± 2.7∗↑	51.2 ± 1.9∗↑	63.0 ± 4.9∗↑
	$CLBP_{24,3}$	85.8 ± 2.7∗	34.5 ± 8.1∗↑	33.2 ± 3.0∗↑	79.2 ± 2.1∗↑	**85.8 ± 1.6**∗↓	71.1 ± 6.5∗↑	78.7 ± 4.3∗↑	84.7 ± 2.1∗↑	62.1 ± 1.4∗↑	**74.7 ± 2.8**∗↓
	$LBC_{24,3}$	64.5 ± 3.0∗	22.5 ± 4.8∗↑	39.3 ± 2.9∗↑	59.9 ± 3.5∗↑	64.0 ± 3.1∗↑	37.7 ± 3.1∗↑	46.9 ± 5.6∗↑	61.8 ± 3.3∗↑	47.5 ± 1.7∗↑	59.7 ± 5.0∗↑
	$CLBC_{24,3}$	70.8 ± 3.2∗	32.3 ± 7.7∗↑	34.4 ± 3.1∗↑	73.8 ± 4.0∗↑	73.4 ± 3.2∗↑	67.1 ± 5.3∗↑	72.3 ± 4.1∗↑	72.8 ± 2.9∗↑	57.2 ± 2.2∗↑	58.3 ± 5.6∗↑
	$DRLBP_{8,1}$	69.7 ± 2.4∗	29.4 ± 7.3∗↑	34.3 ± 3.8∗↑	66.8 ± 3.0∗↑	71.2 ± 3.7∗↑	52.7 ± 5.8∗↑	60.8 ± 6.4∗↑	69.9 ± 3.4∗↑	52.2 ± 1.0∗↑	63.1 ± 3.8∗↑
	EID^{ri}	**92.6 ± 1.1**	**48.9 ± 2.4**	**49.4 ± 1.6**	**87.0 ± 1.7**	84.0 ± 1.4	**81.3 ± 3.0**	**83.0 ± 1.0**	**86.6 ± 1.9**	**70.1 ± 1.2**	72.0 ± 1.9
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∗↑
	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∗
	$LBP^{u2}_{8,1}$	75.5 ± 2.0∗	31.8 ± 3.1∗↑	36.7 ± 4.3∗↑	74.2 ± 2.2∗↑	74.6 ± 2.1∗↑	53.0 ± 7.4∗↑	62.4 ± 4.6∗↑	73.6 ± 3.1∗↑	56.9 ± 1.3∗↑	66.5 ± 3.2∗
	$LBP^{u2ri}_{24,3}$	67.4 ± 2.9∗	20.2 ± 6.5∗↑	38.2 ± 3.3∗↑	66.1 ± 3.2∗↑	66.1 ± 2.6∗↑	40.2 ± 3.7∗↑	50.6 ± 3.7∗↑	64.4 ± 2.8∗↑	53.5 ± 2.7∗↑	66.1 ± 2.9∗
	$CLBP_{24,3}$	**90.6 ± 1.2**−	41.8 ± 2.4∗	33.3 ± 3.5∗↑	37.8 ± 1.3∗↑	**91.0 ± 1.6**−↓	67.5 ± 9.4∗	**77.1 ± 4.0**∗	**90.6 ± 1.6**−↓	63.5 ± 1.0∗↑	**78.7 ± 4.5**∗↓
	$LBC_{24,3}$	66.1 ± 2.8∗	18.4 ± 4.0∗↑	39.8 ± 4.6∗	63.8 ± 3.2∗↑	66.6 ± 2.7∗↑	39.7 ± 5.0∗↑	52.0 ± 4.0∗↑	64.4 ± 3.4∗↑	52.4 ± 3.3∗↑	65.3 ± 2.7∗↑
	$CLBC_{24,3}$	76.7 ± 4.1∗	38.3 ± 3.9∗↑	32.8 ± 3.9∗↑	68.0 ± 3.6∗↑	78.1 ± 3.8∗	64.2 ± 7.4∗↑	69.4 ± 4.5∗↑	77.8 ± 2.5∗↑	59.7 ± 1.6∗↑	51.3 ± 2.4∗↑
	$DRLBP_{8,1}$	86.3 ± 1.1	32.3 ± 6.8∗↑	37.1 ± 3.6∗↑	**85.0 ± 1.5**∗↓	86.9 ± 1.3 ↓	64.3 ± 5.1∗↑	73.0 ± 3.8∗	85.9 ± 1.8 ↓	62.5 ± 1.8∗↑	76.1 ± 3.4∗↓
	EID^{ri}	86.9 ± 1.9	**43.3 ± 1.0**	41.4 ± 1.3	82.4 ± 2.0	80.1 ± 2.0	**70.9 ± 4.1**	75.7 ± 1.7	83.1 ± 2.2	**65.1 ± 1.4**	68.3 ± 1.6
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∗↑
	GLCM	49.0 ± 0.5∗	17.0 ± 2.9∗↑	32.7 ± 2.3∗↑	49.7 ± 0.7∗↑	47.3 ± 0.2∗↑	34.1 ± 2.8∗↑	42.0 ± 2.1∗↑	46.1 ± 0.5∗↑	39.9 ± 1.0∗↑	46.9 ± 1.1∗↑
	$LBP^{u2}_{8,1}$	42.6 ± 1.8∗	19.7 ± 2.3∗↑	25.8 ± 3.3∗↑	39.1 ± 1.8∗↑	45.2 ± 2.1∗↑	27.0 ± 4.6∗↑	38.4 ± 4.2∗↑	44.2 ± 4.7∗↑	32.1 ± 1.2∗↑	37.3 ± 2.6∗↑
	$LBP^{u2ri}_{24,3}$	69.5 ± 3.3∗	20.3 ± 5.2∗↑	**42.9 ± 5.0**∗	67.0 ± 3.7∗↑	66.5 ± 2.6∗↑	43.1 ± 4.8∗↑	51.3 ± 3.5∗↑	66.1 ± 2.5∗↑	54.8 ± 1.6∗↑	66.3 ± 2.3∗↑
	$CLBP_{24,3}$	**89.0 ± 2.8**	40.9 ± 6.3∗	32.4 ± 3.6∗↑	40.1 ± 2.4∗↑	**90.8 ± 1.8**−↓	69.1 ± 8.4∗	**78.1 ± 5.1**∗	**89.5 ± 1.9** ↓	62.1 ± 1.3∗↑	**79.3 ± 3.9**∗↓
	$LBC_{24,3}$	68.3 ± 3.6∗	17.6 ± 4.2∗↑	41.2 ± 4.1∗	65.2 ± 4.0∗↑	64.9 ± 2.9∗↑	41.8 ± 5.0∗↑	50.3 ± 4.2∗↑	64.7 ± 2.9∗↑	52.9 ± 2.0∗↑	64.1 ± 2.9∗↑
	$CLBC_{24,3}$	76.6 ± 3.8∗	37.2 ± 2.1∗↑	30.3 ± 3.0∗↑	66.9 ± 3.3∗↑	77.4 ± 4.4∗↑	65.4 ± 5.4∗↑	69.0 ± 3.6∗↑	75.7 ± 3.8∗↑	59.0 ± 1.6∗↑	56.5 ± 5.8∗↑
	$DRLBP_{8,1}$	74.1 ± 2.1∗	29.1 ± 5.3∗↑	36.0 ± 2.7∗↑	71.9 ± 2.0∗↑	75.7 ± 2.4∗↑	59.9 ± 5.8∗↑	65.2 ± 3.0∗↑	75.4 ± 3.1∗↑	56.0 ± 1.3∗↑	64.9 ± 3.8∗↑
	EID^{ri}	88.6 ± 1.3	**43.0 ± 2.0**	41.2 ± 1.1	**84.4 ± 1.7**	80.5 ± 1.1	**72.4 ± 3.8**	76.5 ± 1.6	84.0 ± 1.4	**65.9 ± 0.9**	68.6 ± 1.8
OutexTC00	DIF	17.8 ± 1.4∗	8.30 ± 2.7∗↑	12.8 ± 4.8∗↑	14.8 ± 1.9∗↑	15.7 ± 1.9∗↑	11.5 ± 2.6∗↑	12.2 ± 4.4∗↑	16.3 ± 1.3∗↑	13.2 ± 1.4∗↑	15.0 ± 2.8∗↑
	GLCM	70.9 ± 2.2∗	20.0 ± 7.3∗↑	36.0 ± 7.9∗↑	60.8 ± 4.8∗↑	66.1 ± 4.9∗↑	41.2 ± 12.5∗↑	52.0 ± 7.1∗↑	69.4 ± 3.0∗↑	49.7 ± 5.4∗↑	70.2 ± 2.8∗↑
	$LBP^{u2}_{8,1}$	**87.9 ± 1.2**	40.8 ± 12.0∗	36.9 ± 3.8∗↑	87.5 ± 1.7	**87.5 ± 1.2** ↓	67.2 ± 10.3∗	73.5 ± 3.1∗↑	**86.8 ± 1.4**	65.6 ± 1.5∗↑	**80.4 ± 3.0**∗↓
	$LBP^{u2ri}_{24,3}$	69.5 ± 2.9∗	26.8 ± 7.9∗↑	39.2 ± 6.1∗↑	67.9 ± 2.4∗↑	66.5 ± 3.3∗↑	44.3 ± 6.9∗↑	51.6 ± 4.2∗↑	68.8 ± 2.2∗↑	53.0 ± 2.1∗↑	69.7 ± 2.8∗↑
	$CLBP_{24,3}$	81.0 ± 2.9∗	37.4 ± 7.6∗↑	23.4 ± 3.6∗↑	27.0 ± 2.2∗↑	82.1 ± 1.3∗↑	61.3 ± 4.2∗↑	68.2 ± 4.3∗↑	80.7 ± 2.5∗↑	56.8 ± 1.3∗↑	70.2 ± 6.5∗
	$LBC_{24,3}$	68.2 ± 3.8∗	20.2 ± 4.6∗↑	36.2 ± 3.8∗↑	68.0 ± 4.1∗↑	63.9 ± 3.7∗↑	39.6 ± 5.8∗↑	48.5 ± 3.0∗↑	66.6 ± 3.9∗↑	50.5 ± 1.7∗↑	66.8 ± 3.0∗↑
	$CLBC_{24,3}$	72.8 ± 2.9∗	37.1 ± 5.8∗↑	30.4 ± 3.6∗↑	69.1 ± 2.9∗↑	73.8 ± 2.9∗↑	64.3 ± 4.9∗↑	66.1 ± 4.9∗↑	71.1 ± 3.5∗↑	57.3 ± 1.5∗↑	55.0 ± 4.7∗↑
	$DRLBP_{8,1}$	85.0 ± 1.8∗	34.1 ± 10.7∗↑	40.9 ± 4.3∗↑	83.4 ± 1.4∗↑	83.7 ± 1.5∗	64.1 ± 5.5∗↑	71.6 ± 3.8∗↑	83.3 ± 1.7∗↑	66.9 ± 1.5∗↑	79.5 ± 4.8∗↓
	EID^{ri}	**87.9 ± 1.8**	**46.0 ± 2.1**	**47.8 ± 2.3**	**87.7 ± 2.2**	84.6 ± 1.7	**72.2 ± 3.6**	**80.1 ± 1.9**	86.0 ± 2.0	**73.3 ± 1.2**	74.5 ± 1.6
OutexTC10	DIF	8.20 ± 0.7∗	6.60 ± 1.5∗↑	11.4 ± 2.1∗↑	7.40 ± 0.7∗↑	8.40 ± 0.7∗↑	7.50 ± 1.6∗↑	7.70 ± 1.7∗↑	9.40 ± 0.7∗↑	9.10 ± 0.8∗↑	7.80 ± 1.2∗↑
	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∗↑
	$LBP^{u2}_{8,1}$	34.4 ± 1.8∗	16.1 ± 3.5∗↑	17.7 ± 4.3∗↑	32.8 ± 1.2∗↑	34.8 ± 2.1∗↑	20.4 ± 3.4∗↑	28.2 ± 4.7∗↑	32.8 ± 1.2∗↑	22.5 ± 0.9∗↑	28.9 ± 4.4∗↑
	$LBP^{u2ri}_{24,3}$	64.5 ± 1.0∗	25.9 ± 5.3∗↑	38.4 ± 4.4∗	64.3 ± 2.0∗↑	64.2 ± 2.4∗↑	34.5 ± 7.0∗↑	49.6 ± 3.3∗↑	61.2 ± 2.8∗↑	47.1 ± 1.2∗↑	62.3 ± 1.7∗↑
	$CLBP_{24,3}$	86.1 ± 2.4	33.8 ± 5.6∗↑	26.6 ± 2.7∗↑	16.1 ± 1.5∗↑	**86.9 ± 2.3** ↓	51.6 ± 7.5∗↑	74.3 ± 6.8∗	**86.5 ± 2.6**	59.0 ± 1.0∗↑	**74.8 ± 5.0**∗
	$LBC_{24,3}$	60.5 ± 1.8∗	20.9 ± 4.3∗↑	32.8 ± 5.7∗↑	59.8 ± 1.7∗↑	59.4 ± 2.7∗↑	33.9 ± 4.1∗↑	45.4 ± 3.2∗↑	55.6 ± 3.6∗↑	44.0 ± 1.5∗↑	57.8 ± 2.4∗↑
	$CLBC_{24,3}$	75.5 ± 2.2∗	31.5 ± 7.0∗↑	29.2 ± 4.0∗↑	70.2 ± 2.3∗↑	77.3 ± 2.5∗↑	59.9 ± 6.9∗↑	71.7 ± 4.6∗↑	75.2 ± 2.4∗↑	58.8 ± 1.9∗↑	53.4 ± 3.9∗↑
	$DRLBP_{8,1}$	64.0 ± 2.6∗	27.2 ± 5.5∗↑	29.8 ± 5.9∗↑	59.4 ± 1.9∗↑	61.3 ± 3.5∗↑	42.2 ± 8.8∗↑	59.3 ± 5.4∗↑	64.7 ± 3.8∗↑	52.0 ± 0.8∗↑	55.4 ± 5.1∗↑
	EID^{ri}	**87.1 ± 1.9**	**41.8 ± 2.5**	**41.2 ± 1.6**	**86.2 ± 1.6**	84.3 ± 1.5	**71.6 ± 4.4**	**78.4 ± 2.2**	86.0 ± 1.9	**68.8 ± 1.6**	72.3 ± 1.3
KySinHw	DIF	11.6 ± 1.1∗	9.00 ± 1.6∗↑	20.5 ± 1.9∗↑	8.20 ± 1.0∗↑	10.9 ± 1.6∗↑	11.5 ± 2.5∗↑	13.2 ± 3.6∗↑	11.9 ± 1.8∗↑	15.8 ± 1.4∗↑	11.6 ± 1.8∗↑
	GLCM	69.6 ± 2.3∗	13.2 ± 3.9∗↑	41.0 ± 3.6∗	67.6 ± 2.6∗↑	64.9 ± 3.0∗↑	51.5 ± 6.8∗↑	61.6 ± 3.2∗↑	67.2 ± 3.0∗↑	55.2 ± 2.0∗↑	67.3 ± 2.3∗↑
	$LBP^{u2}_{8,1}$	54.8 ± 2.2∗	22.8 ± 3.3∗↑	28.2 ± 3.3∗↑	48.6 ± 1.6∗↑	54.8 ± 3.8∗↑	27.1 ± 7.8∗↑	45.0 ± 3.0∗↑	51.5 ± 6.1∗↑	35.0 ± 1.3∗↑	48.4 ± 4.3∗↑
	$LBP^{u2ri}_{24,3}$	81.8 ± 1.5∗	26.1 ± 5.6∗↑	**48.4 ± 3.4**∗↓	78.8 ± 1.6∗↑	81.0 ± 1.7∗↑	46.4 ± 6.5∗↑	60.3 ± 4.2∗↑	78.9 ± 2.3∗↑	63.5 ± 1.5∗↑	78.8 ± 1.7∗↓
	$CLBP_{24,3}$	**97.3 ± 0.7**−	**46.7 ± 4.7**∗	31.6 ± 3.9∗↑	23.1 ± 1.9∗↑	96.5 ± 1.1−↓	74.8 ± 5.4∗↑	83.1 ± 4.1∗	**97.9 ± 1.0**−↓	67.0 ± 1.1∗	**87.3 ± 4.6**∗↓
	$LBC_{24,3}$	80.7 ± 1.5∗	20.6 ± 7.6∗↑	46.3 ± 4.1∗↓	77.7 ± 1.7∗↑	79.5 ± 2.0∗↑	46.5 ± 6.8∗↑	60.2 ± 2.1∗↑	77.5 ± 2.8∗↑	60.6 ± 1.5∗↑	77.6 ± 1.6∗↓
	$CLBC_{24,3}$	89.0 ± 1.9∗	46.2 ± 2.7∗	37.0 ± 4.1∗	87.3 ± 2.0∗↑	89.5 ± 1.5∗↓	73.0 ± 4.4∗↑	78.2 ± 5.1∗↑	89.2 ± 1.9∗↑	**67.6 ± 1.3**∗	69.7 ± 3.5∗↑
	$DRLBP_{8,1}$	85.3 ± 2.3∗	34.9 ± 8.6∗↑	37.8 ± 5.1∗	80.6 ± 2.3∗↑	87.1 ± 2.9∗	55.0 ± 7.9∗↑	72.4 ± 3.8∗↑	86.6 ± 2.6∗↑	62.7 ± 2.0∗↑	77.3 ± 5.2∗
	EID^{ri}	94.1 ± 1.6	44.8 ± 1.8	39.3 ± 1.3	**91.4 ± 2.3**	87.1 ± 1.8	**80.2 ± 3.0**	**84.1 ± 1.5**	92.1 ± 1.4	67.3 ± 1.3	74.8 ± 1.5

EID^{ri} with 1-NN has also significantly outperformed over 96% of the other methods with different classification methods as revealed by the results of the first statistical test. Checking the influence of EID^{ri} on the performance of each classification method compared to that of the other methods shows that EID^{ri} features have significantly improved over 81% of the cases, but also significantly degraded the performance of 11% of the cases. The performances of other 8% cases show either slightly improvement or deterioration when the features of the proposed method are used.

C. Summary

The following observations can be deduced from the results above.

- The proposed method does not require human intervention to automatically evolve a rotation-invariant image descriptor;
- The system does not require domain knowledge and only uses two instances of each class to find a set of good keypoints that can lead to a significantly better performance than using domain-expert designed descriptors in most cases;
- The newly introduced method does not solely detect a set of predefined/designed keypoints such as lines and corners; rather, it automatically designs a set of keypoints and determines how to detect those keypoints;
- The use of EID$^{\text{ri}}$ features have, in the majority of the cases, improved the performance of the different classification methods compared to the use of the other descriptors' features;
- The proposed method dynamically sets the length of the feature vector by automatically selecting the number of the bits in the binary codes, i.e., children of the *code* node;
- Unlike other hand-crafted descriptors, changing the window size does not require changing or redesigning the system since it is handled automatically in EID$^{\text{ri}}$ as the results of the first experiment suggest; and
- The new method can build complicated functions using the combination of simple ones, while it also has the flexibility to use different functions in the function set.

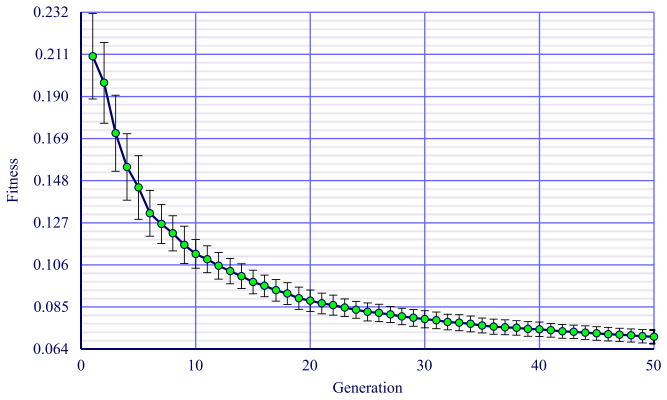


Fig. 15. The average fitness value per generation on the BrWoRo dataset (the whiskers represent the standard deviation).

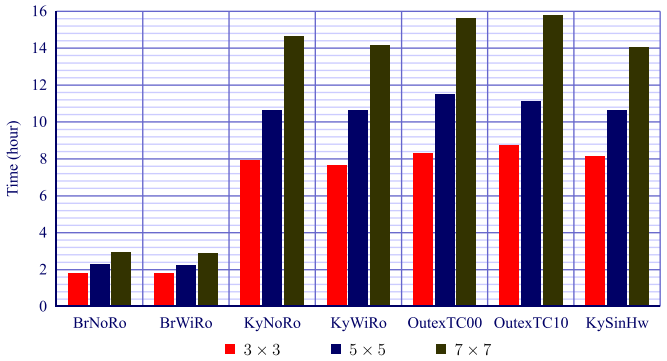


Fig. 16. The impact of the window size on the average time required to evolve a program by EID$^{\text{ri}}$.

VI. Further Analysis

In this section, EID$^{\text{ri}}$ is investigated in depth by considering how and why the proposed method can perform well. The convergence, time required to evolve a descriptor, and number of children of the *code* node and the window size effect on the number of children are discussed in the first subsection. A program evolved by EID$^{\text{ri}}$ is analysed and discussed in details in the second subsection.

A. Overall Analysis

The average fitness value per generation for 30 independent runs using different seed values on two randomly selected instances per class of the BrWoRo dataset is depicted in Fig. 15. A closer inspection of this graph reveals that on average the programs have made larger jumps in the first few generations than the later generations. The fitness value is decreased from 2.10 to 0.088 in the first 20 generations compared to the decrease in fitness from 0.088 to 0.070 over the remainder 30 generations. The standard deviation bars of those 30 independent runs show a similar pattern where the earlier generations have more variations than the later ones.

The average time needed to evolve an image descriptor by EID$^{\text{ri}}$ is presented in Fig. 16. Clearly, the time required to evolve an individual is affected by the number of classes, size of each instance, and size of the sliding window. Having more classes results in having more instances to scan by each GP individual in each generation. Similarly, a larger instance means more pixels are required to be scanned. Although increasing the window size is assumed to reduce the evolutionary time as more pixels (the edges) are ignored, Fig. 16 shows that increasing the window size considerably increases the evolution time. Calculating the minimum, maximum, mean, and standard deviation values require iterations proportional to the number of pixels. This process can be optimised by pre-calculating those values instead of re-calculating them for each individual at each generation. However, storing those pre-calculated values for each instance requires a large amount of memory as each pixel, i.e., a single integer value, will be associated with four floating values.

The evaluation time of EID$^{\text{ri}}$ and the competitive methods have been analysed *theoretically* and *practically* based on [109]. Theoretically, the proposed method has a complexity of $O(n)$, where n is the number of pixels. Similarly, the complexity of the DIF, $\text{LBP}^{u2}_{p,r}$, $\text{LBP}^{u2ri}_{p,r}$, $\text{CLBP}_{p,r}$, $\text{LBC}_{p,r}$, and $\text{CLBC}_{p,r}$ methods are $O(n)$. All those methods need to iterate only once over the pixel values of an image to generate the corresponding feature vector. The $\text{DRLBP}_{p,r}$ method also scans the image only once, however, it performs extra iterations to find the *dominate direction* and performs *feature selection*. GLCM, on the other hand, is a more expensive and computationally intensive method that requires iterating

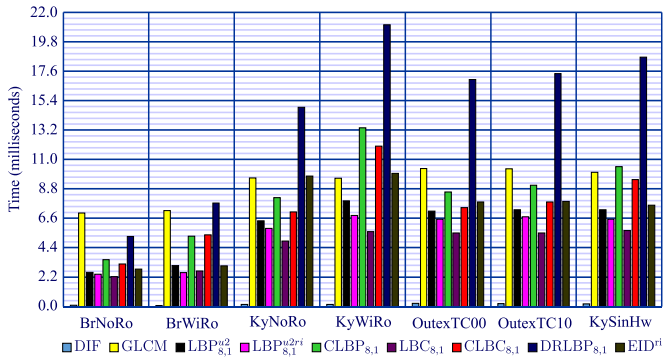


Fig. 17. The average time (milliseconds) to generate the feature vector using different image descriptors on the seven texture datasets.

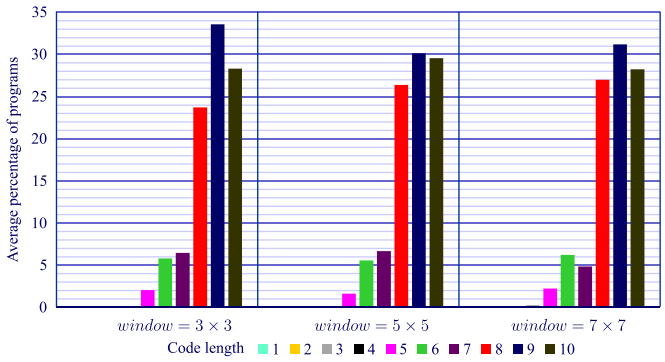


Fig. 18. The average number of children for the *code* node of different window sizes.

multiple times over the image to generate different matrices in order to calculate the Haralick texture features, i.e., feature vector. Practically, further experiments have been conducted to measure the average CPU time required to generate the feature vector for an image (excluding the input/output time to read the image files) as presented in Table IV and Fig. 17. The window size for all the LBP-based methods is set to 3×3 (i.e. $p = 8$ and $r = 1$) for fair comparisons between these methods. The values presented in Table IV were calculated over all instances of each dataset measured in milliseconds. DIF is the fastest amongst all other methods mainly because this method only calculates simple pixel statistics from predefined regions of the image. The results show that EID^{ri} is very fast, requiring on average between 3 and 10 milliseconds to generate the feature vector for an image. Meanwhile, $\text{DRLBP}_{p,r}$ is slower than the other methods as it involves complicated calculations to find the dominant direction and performs feature selection which are computationally intensive.

The average number of children under the *code* node has been analysed as it is automatically determined during the evolutionary process. The bar plot presented in Fig. 18 shows the distribution of code lengths, i.e., number of children for the *code* node, across all independent runs of the seven datasets, and categorised based on the different window sizes. The plot shows clearly that the vast majority of the evolved descriptors have a *code* node with 8, 9 or 10 children. The percentage of those programs with a *code* node having more than 7 children occupy 85.6%, 86.1%, and 86.4% of the evolved programs for window sizes 3×3, 5×5, and 7×7, respectively. Also none of the best evolved programs has less than 3 children under the *code* node, only one program with 3, and only six with 4 nodes. The intuition behind investigating this factor is to give a guideline for future studies on how to set the range of the number of children for the *code* node. Note that the length of the feature vector is doubled for every node added to the children list of the *code* node, which means more space is required to store those feature vectors. Moreover, from the analysis of those programs with 9 or 10 children, it has been observed that only a few cells of the feature vectors have values/counts, whilst the other cells have zero values across all instances. Removing those unused cells can potentially reduces the memory required to store the feature vectors.

B. An Evolved Descriptor

A sample program evolved by EID^{ri} on the BrNoRo dataset is presented in Fig. 19. Using a window of size 5×5 pixels, this program has achieved 92.14% accuracy on the unseen data. Overall, there are 118 nodes in this program where 61 nodes are leaves and the other 57 are functions. The *code* node has 5 children/branches, which means that the feature vector for an image is with a length of $2^5 = 32$ values. Although some of those branches are difficult to interpret, other branches can be simplified and presented as mathematical formulae. For instance, the corresponding formulae for the third, fourth and fifth branches can be presented as $2mean - (min + max)$, $(2mean - max) \,/\, (stdev - max)$, and $\left(mean + max + \frac{mean - min}{stdev}\right) - \left((mean - min)\,\frac{stdev}{max}\right)$, respectively. As only a few simple mathematical operations are required to generate the feature vector for an image, such descriptors can be used for on-line applications.

Table V presents the confusion matrix for the program depicted in Fig. 19. The first column of this table lists the actual class labels, whereas the first row lists the predicted class labels. The proportion of correctly classified instances for each class is presented in the last column of Table V. The program has correctly classified either all instances (100% accuracy) or achieved over 90% accuracy for 17 out of the 20 classes, scored over 70% accuracy on 2 further classes, and did not perform well on only one class (D09) with 55% accuracy. The confusion matrix shows that 19 out of 42 instances of the D09 class have been misclassified as D04. A sample instance from each of these two classes, i.e., D04 and D09, are randomly selected and two tiles of each are enlarged (zoomed-in) in Fig. 20. A close look at the enlarged tiles reveals the similarity between the instances of the two classes especially when the rotation variation is considered (the position order of the pixel values is ignored within the sliding window).

The program presented in Fig. 19 is further analysed by feeding two randomly selected instances from each class of the BrNoRo dataset to generate the corresponding feature vectors that are visually depicted in Fig. 21. In this figure, the two instances drawn from each class are positioned side-by-side to ease the comparison task, and the corresponding class label is shown on the horizontal axis. The values of each feature vector are normalised and represented as the

TABLE IV
THE AVERAGE TIME (MILLISECONDS) TO GENERATE THE FEATURE VECTOR FOR AN IMAGE USING NINE IMAGE DESCRIPTORS ON SEVEN DATASETS.

	DIF	GLCM	$\text{LBP}^{u2}_{8,1}$	$\text{LBP}^{u2ri}_{8,1}$	$\text{CLBP}_{8,1}$	$\text{LBC}_{8,1}$	$\text{CLBC}_{8,1}$	$\text{DRLBP}_{8,1}$	EID^{ri}
BrNoRo	0.090	6.992	2.545	2.408	3.502	2.222	3.180	5.233	2.796
BrWiRo	0.082	7.170	3.059	2.527	5.250	2.663	5.352	7.734	3.043
KyNoRo	0.148	9.605	6.397	5.832	8.147	4.890	7.068	14.902	9.755
KyWiRo	0.148	9.587	7.901	6.806	13.370	5.604	12.001	21.066	9.960
OutexTC00	0.230	10.321	7.127	6.535	8.560	5.506	7.398	16.973	7.823
OutexTC10	0.207	10.306	7.232	6.702	9.070	5.495	7.825	17.428	7.851
KySinHw	0.190	10.033	7.234	6.535	10.469	5.671	9.495	18.662	7.572

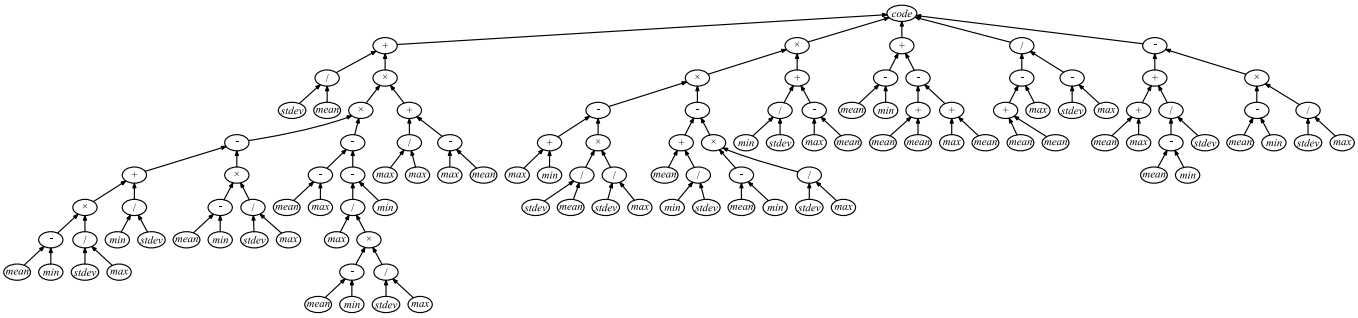
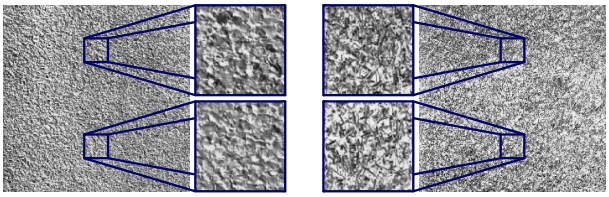


Fig. 19. A sample program evolved by EID^{ri} on the BrNoRo dataset (5 bits).

TABLE V
THE CONFUSION MATRIX FOR THE PROGRAM PRESENTED IN FIG. 19

	D01	D03	D04	D05	D06	D09	D11	D14	D15	D16	D17	D18	D20	D21	D24	D34	D37	D46	D47	D49	
D01	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0.91
D03	0	40	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0.95
D04	0	0	30	0	0	9	2	0	0	0	0	0	0	0	1	0	0	0	0	0	0.71
D05	0	0	0	40	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0.95
D06	0	0	0	0	42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00
D09	0	0	19	0	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.55
D11	0	0	2	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0.95
D14	0	1	0	0	0	0	0	41	0	0	0	0	0	0	0	0	0	0	0	0	0.98
D15	0	0	2	0	0	0	0	0	38	0	0	0	0	0	2	0	0	0	0	0	0.91
D16	0	0	0	0	0	0	0	0	0	39	2	0	0	1	0	0	0	0	0	0	0.93
D17	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	0	0	0	0	1.00
D18	0	0	0	9	0	0	0	0	0	0	0	31	0	0	0	0	2	0	0	0	0.74
D20	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	0	0	1.00
D21	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	0	1.00
D24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	1.00
D34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	1.00
D37	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	38	1	0	0	0.91
D46	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	40	0	0	0.95
D47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	1.00
D49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	1.00



(a) (b)

Fig. 20. A sample instance and two enlarged tiles of the (a) D04 class, and (b) D09 class.

percentage that are placed on top of each other. The $2^5 = 32$ features $\{f_1, f_2, \ldots, f_{32}\}$ are indicated using different colours as depicted in the legend of Fig. 21. The figure clearly shows that the evolved program has generated a "fingerprint" for the instances belonging to the same class that is different from all other instances from the other classes. Some examples are the class D16 that has a very high percentage of f_3, class D21 with a high percentage of f_9, and class D49 with high f_{31} percentage. The program also identified some features that occur in some classes but not in others, e.g., feature f_{12} that does not appear in the vectors of classes D01, D03, D06, D14, D16, D21, D46, and D49. The similarity between D04 and D09 instances is also noticeably high, which explains why a large number of instances of these two classes were misclassified.

VII. CONCLUSIONS

In this paper, GP has been successfully utilised to automatically synthesise a set of mathematical formulae that form an image descriptor. Unlike other image descriptors, the proposed method does not require human intervention to design a set of keypoints, or any mechanism for detecting those keypoints and extracting the feature values. Domain knowledge is not required either. The proposed method uses two instances of each class to evolve a descriptor that is capable of generating distinctive feature vectors for instances belong to different classes. As this method uses only a few training instances, it is suitable for problems where the number of labelled data is limited. Another impact of using a few training instances is that the overall complexity of the system in terms of time and physical computer memory will be reduced, which makes the system suitable for the situations that cannot afford a long time for training. Moreover, the proposed method uses a set of first-order statistics to evolve a rotation-invariant descriptor relying on the order-independent characteristic of those statistics. An evolved descriptor works in a pixel-by-pixel fashion, using a sliding window of a predefined size. To assess the effectiveness of the proposed method, seven texture image classification datasets with different degrees of rotations were used and compared to the effectiveness of eight state-of-the-art expert-designed and hand-crafted image descriptors using ten widely used classification algorithms. Quantitatively, the results of

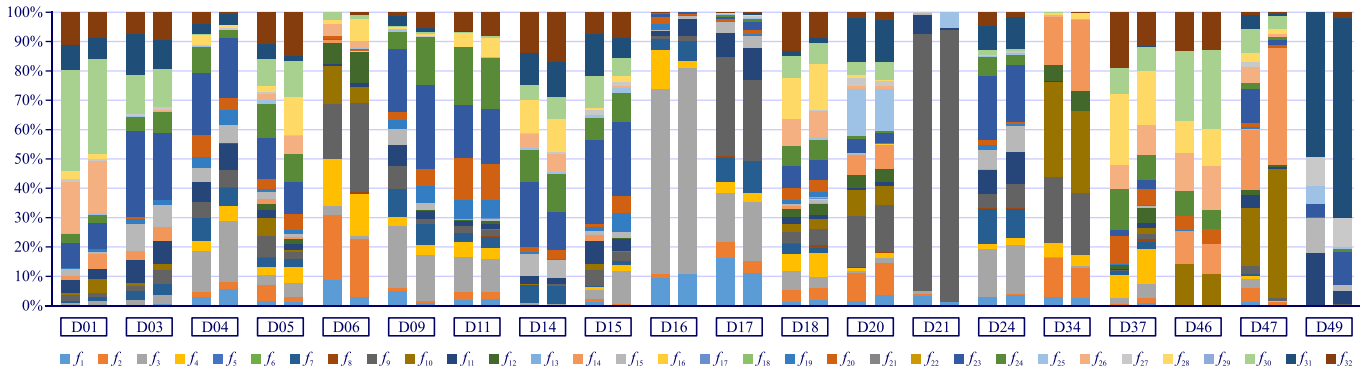


Fig. 21. A visual representation for the feature vectors of 40 randomly selected instances from the 20 classes (two from each class) of the BrNoRo dataset, which were generated by the program presented in Fig. 19.

the experiments reveal the effectiveness of the proposed GP method to evolve an image descriptor using only two instances per class and yet can significantly outperform or achieve comparable results to the hand-crafted descriptors. Qualitatively, the analysis of the proposed system and an evolved descriptor demonstrates the robustness to tackle the rotation variation and the interpretability of the evolved descriptor.

A. Major Contributions

The major contribution of the method proposed in this paper is how GP can be designed to have a dynamic program representation that allows the system to specify the length of the feature vector simultaneously while evolving an image descriptor. The empirical evaluations reveal the effectiveness of the proposed method to consider different aspects, and yet, is capable of evolving an illumination- and rotation-invariant image descriptor without domain-knowledge, human intervention, and using only two instances per class.

B. Future Work

Although the proposed method has tackled the issues of hand-crafted descriptors as well as those recently proposed methods, i.e., GP-criptor [45] and EID [46], it has some limitations and there is still room for improvements that will be addressed in the future. The proposed method can evolve illumination-invariant and rotation-invariant descriptors, but it may fail to handle the scale variation. Building a scale-invariant descriptor is more difficult and more challenging than illumination and rotation image variants. One way to tackle this problem is via using windows of different sizes in order to capture keypoints at different levels (zoom). This could be accomplished by updating the program representation to use different window sizes during the evolutionary process. Another limitation is that dense descriptors have the potential to perform very well on texture images, which may not be as good as sparse descriptors when it comes to general object classification. In real-life object classification tasks, it is more likely that the objects of a category will have different backgrounds, colours, and appearances. Increasing the number of training instances can be adopted to mitigate this problem, which may be achieved via transfer learning [110]–[112]. Using a set of instances from a related (source) domain to evolve a model that will be used to solve the problem at hand (target domain) is one way of performing transfer learning that we would like to investigate in the future. Colours are substantial elements that carry more information than grey-scale images, and can largely influence the performance of the different tasks in computer vision applications such as content-based image retrieval and salient object detection. Hence, in the future we also would like to extend the proposed method to handle colours and evolve colour image descriptors.

ACKNOWLEDGMENT

This work was supported in part by the Marsden Fund of New Zealand Government under Contracts VUW1209 and VUW1509, the University Research Fund of Victoria University of Wellington under contracts 209861/3580, 209862/3580 and 213150/3663, and the Huawei Program under grant number E2880/3663.

REFERENCES

[1] X. Zhu, C. Vondrick, C. C. Fowlkes, and D. Ramanan, "Do we need more training data?" *International Journal of Computer Vision*, vol. 119, no. 1, pp. 76–92, 2016.

[2] B. Xue, M. Zhang, W. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016.

[3] M. Pechenizkiy, "The impact of feature extraction on the performance of a classifier: kNN, Naïve Bayes and C4.5," in *Proceedings of the 18th Conference of the Canadian Society for Computational Studies of Intelligence: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, vol. 3501. Springer, 2005, pp. 268–279.

[4] H. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Robotics Institute, Stanford University, Tech. Rep. CMU-RI-TR-80-03, 1980.

[5] 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.

[6] A. Willis and Y. Sui, "An algebraic model for fast corner detection," in *Proceedings of the 12th IEEE International Conference on Computer Vision*. IEEE, 2009, pp. 2296–2302.

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

[8] T. Lindeberg, "Feature detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, no. 2, pp. 79–116, 1998.

[9] 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.

[10] 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.

[11] S. Krig, *Computer Vision Metrics: Survey, Taxonomy, and Analysis*, 1st ed. Apress, 2014.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] J. Chen, G. Zhao, V. Kellokumpu, and M. Pietikäinen, "Combining sparse and dense descriptors with temporal semantic structures for robust human action recognition," in *IEEE International Conference on Computer Vision Workshops*. IEEE, 2011, pp. 1524–1531.

[18] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on feature distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[19] H. Al-Sahaf, A. Al-Sahaf, B. Xue, M. Johnston, and M. Zhang, "Automatically evolving rotation-invariant texture image descriptors by genetic programming," *IEEE Transactions on Evolutionary Computation*, pp. 1–19, 2016, doi:10.1109/TEVC.2016.2577548.

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

[21] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*. Published via http://lulu.com, 2008, (with contributions by J. R. Koza).

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] T. Liddle, M. Johnston, and M. Zhang, "Multi-objective genetic programming for object detection," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[28] H. Vojodi, A. Fakhari, and A. M. E. Moghadam, "A new evaluation measure for color image segmentation based on genetic programming approach," *Image and Vision Computing*, vol. 31, no. 11, pp. 877–886, 2013.

[29] Y. Liang, M. Zhang, and W. Browne, "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.

[30] W. B. Langdon, M. Modat, J. Petke, and M. Harman, "Improving 3D medical image registration CUDA software with genetic programming," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2014, pp. 951–958.

[31] S. Harding, J. Leitner, and J. Schmidhuber, "Cartesian genetic programming for image processing," in *Genetic Programming Theory and Practice X*, ser. Genetic and Evolutionary Computation. Springer, 2013, pp. 31–44.

[32] 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.

[33] 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.

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

[35] 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.

[36] 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.

[37] 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.

[38] 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. 46, no. 1, pp. 158–172, 2016.

[39] S. J. Raudys and A. K. Jain, "Small sample size effects in statistical pattern recognition: Recommendations for practitioners," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 252–264, 1991.

[40] G. M. Weiss and F. Provost, "Learning when training data are costly: The effect of class distribution on tree induction," *Journal of Artificial Intelligence Research*, vol. 19, no. 1, pp. 315–354, 2003.

[41] E. Rodner and J. Denzler, "Learning with few examples by transferring feature relevance," in *Proceedings of the 31st DAGM Symposium on Pattern Recognition*, ser. Lecture Notes in Computer Science, vol. 5748. Springer, 2009, pp. 252–261.

[42] L. Zhuang, H. Gao, J. Luo, and Z. Lin, "Regularized semi-supervised latent dirichlet allocation for visual concept learning," *Neurocomputing*, vol. 119, pp. 26–32, 2013.

[43] X. Zhu, C. Vondrick, D. Ramanan, and C. Fowlkes, "Do we need more training data or better models for object detection?" in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2012, pp. 80.1–80.11.

[44] H. Al-Sahaf, M. Zhang, and M. Johnston, "Binary image classification: A genetic programming approach to the problem of limited training instances," *Evolutionary Computation (Journal, MIT Press)*, vol. 24, no. 1, pp. 143–182, 2016.

[45] 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.

[46] H. Al-Sahaf, M. Zhang, and M. Johnston, "Evolutionary image descriptor: A dynamic genetic programming representation for feature extraction," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 975–982.

[47] 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.

[48] 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.

[49] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face description with local binary patterns: Application to face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 2037–2041, 2006.

[50] 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.

[51] D. T. Nguyen, Z. Zong, P. Ogunbona, and W. Li, "Object detection using non-redundant local binary patterns," in *Proceedings of the 17th IEEE International Conference on Image Processing*. IEEE, 2010, pp. 4609–4612.

[52] 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.

[53] J. Cheng, L. Li, B. Luo, S. Wang, and H. Liu, "High-resolution remote sensing image segmentation based on improved RIU-LBP and SRM," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, pp. 1–12, 2013.

[54] 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.

[55] O. A. Vatamanu, M. Frandes, D. Lungeanu, and G.-I. Mihalas, "Content based image retrieval using local binary pattern operator and data mining techniques," *Studies in Health Technology and Informatics*, vol. 210, pp. 75–79, 2015.

[56] 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, no. 1842. Springer, 2000, pp. 404–420.

[57] A. Hafiane, G. Seetharaman, and B. Zavidovique, "Median binary pattern for textures classification," in *Image Analysis and Recognition*, ser. Lecture Notes in Computer Science. Springer, 2007, vol. 4633, pp. 387–398.

[58] T. Ahonen and M. Pietikäinen, "Soft histograms for local binary patterns," in *Proceedings of the Finnish Signal Processing Symposium*. Oulu, 2007, pp. 1–4.

[59] D. Iakovidis, E. Keramidas, and D. Maroulis, "Fuzzy local binary patterns for ultrasound texture characterization," in *Image Analysis and Recognition*, ser. Lecture Notes in Computer Science, A. Campilho and M. Kamel, Eds. Springer, 2008, vol. 5112, pp. 750–759.

[60] X. Tan and B. Triggs, "Enhanced local texture feature sets for face recognition under difficult lighting conditions," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1635–1650, 2010.

[61] L. Nanni, A. Lumini, and S. Brahnam, "Local binary patterns variants as texture descriptors for medical image analysis," *Artificial Intelligence in Medicine*, vol. 49, no. 2, pp. 117–125, 2010.

[62] X. Qian, X.-S. Hua, P. Chen, and L. Ke, "PLBP: An effective local binary patterns texture descriptor with pyramid representation," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2502–2515, 2011.

[63] 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.

[64] Y. Zhao, W. Jia, R.-X. Hu, and H. Min, "Completed robust local binary pattern for texture classification," *Neurocomputing*, vol. 106, no. 1, pp. 68–76, 2013.

[65] R. Mehta and K. Egiazarian, "Dominant rotated local binary patterns (DRLBP) for texture classification," *Pattern Recognition Letters*, vol. 71, no. 1, pp. 16–22, 2016.

[66] M. Pietikäinen, A. Hadid, G. Zhao, and T. Ahonen, "Local binary patterns for still images," in *Computer Vision Using Local Binary Patterns*, ser. Computational Imaging and Vision. Springer, 2011, vol. 40, pp. 13–47.

[67] G. Kylberg and I.-M. Sintorn, "Evaluation of noise robustness for local binary pattern descriptors in texture classification," *EURASIP Journal on Image and Video Processing*, vol. 2013, no. 1, pp. 1–20, 2013.

[68] J. Santamaria, S. Damas, O. Cordon, and A. Escamez, "Self-adaptive evolution toward new parameter free image registration methods," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 4, pp. 545–557, 2013.

[69] G. Abo Smara and F. Khalefah, "Localization of license plate number using dynamic image processing techniques and genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 244–257, 2014.

[70] M. Lones, S. Smith, J. Alty, S. Lacy, K. Possin, D. Jamieson, and A. Tyrrell, "Evolving classifiers to recognize the movement characteristics of parkinson's disease patients," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 559–576, 2014.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

[76] 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.

[77] 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.

[78] U. Bhowan, M. Johnston, M. Zhang, and X. Yao, "Reusing genetic programming for ensemble selection in classification of unbalanced data," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 6, pp. 893–908, 2014.

[79] ——, "Evolving diverse ensembles using genetic programming for classification with unbalanced data," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 368–386, 2013.

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

[81] 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.

[82] H. Al-Sahaf, K. Neshatian, and M. Zhang, "Two-tier genetic programming for automatic feature extraction, feature selection and image classification," in *Proceedings of the 26th International Conference on Image and Vision Computing New Zealand*. IEEE, 2011, pp. 109–114.

[83] ——, "Automatic feature extraction and image classification using genetic programming," in *Proceedings of the 5th International Conference on Automation, Robots and Applications*. IEEE Press, 2011, pp. 157–162.

[84] S. Hindmarsh, P. Andreae, and M. Zhang, "Genetic programming for improving image descriptors generated using the scale-invariant feature transform," in *Proceedings of the 27th International Conference on Image and Vision Computing New Zealand*. ACM, 2012, pp. 85–90.

[85] C. Ryan, J. Fitzgerald, K. Krawiec, and D. Medernach, "Image classification with genetic programming: Building a stage 1 computer aided detector for breast cancer," in *Handbook of Genetic Programming Applications*. Springer, 2015, pp. 245–287.

[86] W. Fu, M. Johnston, and M. Zhang, "Automatic construction of invariant features using genetic programming for edge detection," in *Proceedings of the 25th Australasian Joint Conference on Artificial Intelligence*, ser. Lecture Notes in Computer Science. Springer, 2012, vol. 7691, pp. 144–155.

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

[88] I. Sobel and G. Feldman, "A 3×3 isotropic gradient operator for image processing," 1968, Presented at a talk at the Stanford Artificial Project.

[89] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[90] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd ed. Springer-Verlag, 1994.

[91] Q. Chen, M. Zhang, and B. Xue, "Feature selection to improve generalisation of genetic programming for high-dimensional symbolic regression," *IEEE Transactions on Evolutionary Computation*, 2017.

[92] 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.

[93] 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.

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

[95] 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.

[96] G. Kylberg, "Automatic virus identification using TEM: Image segmentation and texture analysis," Ph.D. dissertation, Division of Visual Information and Interaction, Uppsala University, Uppsala, Sweden, 2014.

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

[98] S. Luke and L. Spector, "A revised comparison of crossover and mutation in genetic programming," in *Proceedings of the 12th Conference on Genetic Programming*. Morgan Kaufmann, 1998, pp. 208–213.
[99] D. White and S. Poulding, "A rigorous evaluation of crossover and mutation in genetic programming," in *Proceedings of the 12th European Conference on Genetic Programming*. Springer, 2009, pp. 220–231.
[100] 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.
[101] 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.
[102] 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.
[103] 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.
[104] S. Luke, *Essentials of Metaheuristics*, 2nd ed. Lulu, 2013, [Online] Available: http://cs.gmu.edu/~sean/book/metaheuristics/.
[105] 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.
[106] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
[107] 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.
[108] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.
[109] O. A. B. Penatti, E. Valle, and R. S. Torres, "Comparative study of global color and texture descriptors for web image retrieval," *Journal of Visual Communication and Image Representation*, vol. 23, no. 2, pp. 359–380, 2012.
[110] 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.
[111] 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*. Omnipress, 2010, pp. 151–158.
[112] M. Iqbal, B. Xue, H. Al-Sahaf, and M. Zhang, "Cross-domain reuse of extracted knowledge in genetic programming for image classification," *IEEE Transactions on Evolutionary Computation*, 2017, doi:10.1109/TEVC.2017.2657556.

Harith Al-Sahaf (M'13) received his B.Sc. degree in Computer Science from Baghdad University, Baghdad, Iraq, in 2005. He received his Master of Computer Science (MCompSc) degree, and PhD in computer science degree, respectively, in 2010 and 2017 from Victoria University of Wellington, Wellington, New Zealand (VUW). Since 2010, he has joined the Evolutionary Computation Research Group (ECRG) at VUW. Currently, he is a postdoctoral fellow with the School of Engineering and Computer Science at VUW.

Harith's research interests are in evolutionary computation, computer vision, pattern recognition, machine learning, feature manipulation including feature detection, selection, extraction and construction, transfer learning, domain adaptation, one-shot learning, and image understanding.

Harith is a member of the IEEE Computational Intelligence Society (CIS). He is also a member of the IEEE CIS Task Force on Evolutionary Computation for Feature Selection and Construction. He has been serving as a reviewer for top international journals and conferences in the field.

Mengjie Zhang (M'04-SM'10) received the B.E. and M.E. degrees from Artificial Intelligence Research Center, Agricultural University of Hebei, Hebei, China, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 1989, 1992, and 2000, respectively.

Since 2000, he has been with the Victoria University of Wellington, New Zealand, where he is currently Professor of Computer Science, Head of the Evolutionary Computation Research Group, and the Associate Dean (Research and Innovation) in the Faculty of Engineering. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, feature selection and dimensionality reduction, multiobjective optimization, classification with missing data, and job shop scheduling. He has published over 400 academic papers in refereed international journals and conferences.

Prof. Zhang has been serving as an Associated Editor or Editorial Board Member for eight international journals (including IEEE Transactions on Evolutionary Computation, Evolutionary Computation Journal) and as a Reviewer of over 20 international journals. He has been serving as a Steering Committee Member and a Program Committee Member for over 80 international conferences. He has supervised over 50 postgraduate research students. He is the Chair of the IEEE CIS Emergent Technologies Technical Committee, the immediate past Chair of the Evolutionary Computation Technical Committee, a member of the IEEE CIS Intelligent Systems and Applications Technical Committee, a Vice-Chair of the IEEE CIS Task Force on Evolutionary Computer Vision and Image Processing, a Vice-Chair of the IEEE CIS Task Force on Evolutionary Computation for Feature Selection and Construction, a member of IEEE CIS Task Force of Hyper-heuristics, and the Founding Chair for IEEE Computational Intelligence Chapter in New Zealand.

Ausama Al-Sahaf (S'16) received his B.Sc. degree in Computer Science from Al-Mamon University College, Baghdad, Iraq, in 2001. He joined the College of Agriculture, Baghdad University, Baghdad, Iraq, as a faculty member in September 2002. He received his Master of Computer Science (MCompSc) degree in 2010 from Victoria University of Wellington, Wellington, New Zealand (VUW).

Ausama's research interests are in evolutionary computation, computer vision, pattern recognition, machine learning, and transfer learning. Ausama is a member of the IEEE Computational Intelligence Society (CIS).

Mr. Al-Sahaf has worked as a software developer in both governmental and private sector companies.

Mark Johnston (M'10) is currently a Senior Lecturer in Mathematics at the University of Worcester, UK, having previously held positions at Victoria University of Wellington, New Zealand, and the University of Essex, UK. He received a BSc(hons) in Mathematics and a PhD in Operations Research from Massey University, New Zealand.

Mark's research interests are in combinatorial optimisation, particularly evolutionary computation approaches and hyperheuristics. For several years Mark has worked closely with the New Zealand Rugby Union in developing their domestic rugby competition schedules.