A Multitree Genetic Programming Representation for Automatically Evolving Texture Image Descriptors

Harith Al-Sahaf(✉), Bing Xue, and Mengjie Zhang

School of Engineering and Computer Science, Victoria University of Wellington, P.O. Box 600, Wellington 6140, New Zealand
{harith.al-sahaf,bing.xue,mengjie.zhang}@ecs.vuw.ac.nz

Abstract. Image descriptors are very important components in computer vision and pattern recognition that play critical roles in a wide range of applications. The main task of an image descriptor is to automatically detect micro-patterns in an image and generate a feature vector. A domain expert is often needed to undertake the process of developing an image descriptor. However, such an expert, in many cases, is difficult to find or expensive to employ. In this paper, a multitree genetic programming representation is adopted to automatically evolve image descriptors. Unlike existing hand-crafted image descriptors, the proposed method does not rely on predetermined features, instead, it automatically identifies a set of features using a few instances of each class. The performance of the proposed method is assessed using seven benchmark texture classification datasets and compared to seven state-of-the-art methods. The results show that the new method has significantly outperformed its counterpart methods in most cases.

Keywords: Multitree · Image classification · Feature extraction

1 Introduction

Discriminating between texture images is highly dependent on the detected micro-patterns, i.e., keypoints, such as lines, spots and homogeneous regions presented in those images. Designing a method to automatically identify or detect such micro-patterns is often require human intervention to carry out this task. The detection can be performed either manually, where a domain expert highlights the coordinates of those keypoints, or automatically by using keypoint detectors such as corner detection [21] and local binary patterns (LBP) [18]. Over the past 50 years, image descriptors have emerged to automatically detect a set of predetermined micro-patterns in order to extract the feature vector for an image [11]. The majority of those image descriptors have two limitations. Firstly, they are designed to detect a specific set of micro-patterns such as corners; secondly, they are hand-crafted where domain expert intervention is needed to design and develop those descriptors.

Y. Shi et al. (Eds.): SEAL 2017, LNCS 10593, pp. 499–511, 2017.
https://doi.org/10.1007/978-3-319-68759-9_41

Genetic Programming (GP) is an evolutionary computation technique that mimics the principles of natural selection and survival of the fittest, where a population of computer programs are evolved over generations to find a solution for a user-defined problem [10].

Conventionally, a GP individual is represented by tree structure, where the leaf nodes are populated from the *terminal* set and the internal nodes are populated from the *function* set. The tree-based representation is one of the most commonly used individual representations in GP [20], but it is not the only one. Over the past 30 years, different individual representations have been investigated such as linear GP, multitree GP, and cartesian GP [20].

Automatically evolving interest point detectors by utilising GP has been proposed in [7], and further studied by Olague and Trujillo [19].

Fu *et al.* [8] proposed a GP method to construct invariant features for edge detection. In order to improve the extracted features from raw pixel values, the distributions of the observations from GP programs are used. Their results show that the constructed features by GP with distribution estimation have improved the detection performance compared to the combination of linear support vector machine and a Bayesian model.

Cordelia *et al.* [6] proposed a multitree based GP method for generating prototypes in classification problems, where a dynamic representation is used to allow each individual to have different number of trees. This dynamic representation allows the method to cope with situation where one or more classes comprise subclasses. Using three well-known datasets and compared to another GP based method, their method has achieved significantly better performance as shown in [6].

Broic and Estevez [3] utilised multitree GP and information theory to perform clustering. In their method, an information theory based fitness function is developed to measure the goodness of an evolved program. Moreover, probabilistic based interpretation of the trees' output is used in order to avoid the requirement for a *conflict resolution* phase. The results of their experiments show the superiority of this method compared to k-means clustering using 10 clustering benchmark datasets.

Utilising multitree GP to automatically discover some patterns for self-assembling swarm robots is proposed in [14]. Promising results have been achieved by this method, which reflect its effectiveness.

Recently, Al-Sahaf *et al.* [1] utilised GP to automatically evolve LBP-like rotation-invariant image descriptors using a set of arithmetic operators, first-order statistics and a special *code* node. Strongly-typed GP (STGP) [16] is required in order to specify the structure of an evolved program by this method. Their results reveal the ability of the automatically evolved descriptors to outperform their counterpart hand-crafted descriptors.

The proposed method in [1] represents the baseline for the newly introduced method in this paper, where the representation of an evolved program is the main difference between the two methods.

The overall goal of this study is to utilise multitree GP to the task of automatically evolving rotation-invariant image descriptors. The proposed method uses simple arithmetic operators and first-order statistics, and only two instances per class to build a GP program that scans an image using a sliding window to generate the feature vector. Specifically, this study aims at providing answers for the following questions.

- How a descriptor can be represented using a multitree GP representation?
- What fitness function can be used when there are only a few instances per class in the training set?
- Is the proposed method able to evolve image descriptors that can outperform the hand-crafted descriptors?

2 Background

The baseline method *rotation-invariant GP descriptor* (GP-criptor$^{\text{ri}}$) [1] is discussed in this section.

GP-criptor$^{\text{ri}}$ is a GP based method that aims at automatically evolving rotation-invariant image descriptors using only two instances per class. GP-criptor$^{\text{ri}}$ uses a tree based representation, where each individual is represented by a single tree. The function set comprises the four arithmetic operators $+$, $-$, $\times$ and $/$, and a special *code* node type. Apart from *code*, these functions takes two arguments and they have their corresponding arithmetic meaning. A *code* node takes a predefined number of children and returns a binary code by substituting the value returned by each of its children by 0 if it is negative, and 1 otherwise. The terminal set in GP-criptor$^{\text{ri}}$ consists of four node types $min(\cdot)$, $max(\cdot)$, $mean(\cdot)$, and $stdev(\cdot)$ each of which operates on a vector of values and returns the minimum, maximum, mean and standard deviation, respectively. The order-invariant property of these four operators allowed GP-criptor$^{\text{ri}}$ to evolve rotation-invariant image descriptors [1].

The distances of between-class and within-class are used in the fitness function of GP-criptor$^{\text{ri}}$ in order to allow the method to cope with having a small number of training instances. The fitness function of GP-criptor$^{\text{ri}}$ is defined as:

$$fitness' = 1 - \left(1 \Big/ \left(1 + e^{-5\left(D'_{\text{b}} - D'_{\text{w}}\right)}\right)\right) \tag{1}$$

where D'_{b} and D'_{w} are, respectively, the average distance of between-class and the average distance of within-class. These two distances are defined as:

$$D'_{\text{b}} = \frac{1}{z\,(z-n)} \sum_{\mathbf{u}\in\mathbf{R}} \sum_{\mathbf{v}\in\mathbf{R}\setminus\mathbf{u}} \chi^2\left(\boldsymbol{u},\boldsymbol{v}\right), \quad \{\boldsymbol{u}\in\mathbf{u},\boldsymbol{v}\in\mathbf{v}\} \tag{2}$$

$$D'_{\text{w}} = \frac{1}{z\,(n-1)} \sum_{\mathbf{u}\in\mathbf{R}} \sum_{\boldsymbol{u},\boldsymbol{v}\in\mathbf{u}} \chi^2\left(\boldsymbol{u},\boldsymbol{v}\right), \quad \{\boldsymbol{u}\neq\boldsymbol{v}\} \tag{3}$$

where z is the total number of instances in the training set, n is the number of instances per class, and $\mathbf{R} = \{(\boldsymbol{v}_i, \ell_i)\}$ is the training set. $\boldsymbol{v}_i$ denotes the feature

vector of the i^{th} instance and the corresponding class label is denoted by ℓ_i, where $\boldsymbol{v}_i \in \mathbb{R}_{\geq 0}$, $\ell_i \in \{1, 2, \ldots, c\}$, c is the number of classes, and $i \in \{1, 2, \ldots, z\}$. The distance between two feature vectors is measured by the widely used *Chi-square* (χ^2) measure [5], which is defined as:

$$\chi^2(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2} \sum_{i=1}^{E} \frac{(\boldsymbol{u}_i - \boldsymbol{v}_i)^2}{(\boldsymbol{u}_i + \boldsymbol{v}_i)} \tag{4}$$

where E is the number of elements, $\boldsymbol{u}$ and $\boldsymbol{v}$ are two feature vectors of the same length, i.e., consist of equal umber of elements, and $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are, respectively, the i^{th} element in $\boldsymbol{u}$ and $\boldsymbol{v}$.

3 The Proposed Method

The proposed *multitree GP rotation-invariant image descriptor* ($\text{MGPD}^{\text{ri}}_{t,w}$) method is explained in this section.

3.1 Overall Algorithm

Similar to other machine learning methods, the overall algorithm can be divided into five parts as depicted in Fig. 1. In the first part (dataset preprocessing), the system divides the dataset equally into two subsets each of which comprises 50% of the total instances in each class. The system *randomly* selects two instances of each class from the first subset to form the training set ($\mathbf{S_{tr}}$); whereas the second subset will be used for evaluating the performance of the system, i.e., the test set ($\mathbf{S_{ts}}$). In the second part (image descriptor evolution), the system feeds the training set ($\mathbf{S_{tr}}$) into GP to evolve an image descriptor. The evolved descriptor is then used to transform the training and test sets, i.e., generates the feature vector for each image in the two sets, which is the third part of the overall algorithm (dataset transformation). The transformed training and test sets are, respectively, denoted as $\mathbf{R}$ and $\mathbf{S}$. The fourth part (building a classifier) concerns with building a classifier by feeding the transformed training set ($\mathbf{R}$) into a classification algorithm. The fifth and final part of the overall algorithm (evaluation) uses the transformed test set ($\mathbf{S}$) and the built classifier in order to assess the goodness of the evolved descriptor to generate feature vectors that are sufficient to be classified by the built classifier. More details regarding these five parts are provided in the following subsections.

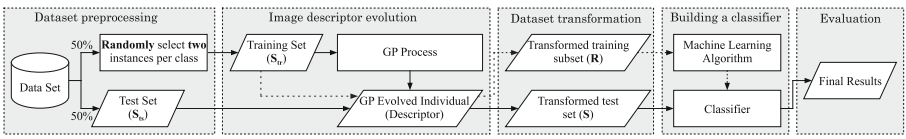

Fig. 1. The overall algorithm of the proposed method.

3.2 Program Representation

Here, a tree-based GP representation [20] is used, where a tree is a set of connected nodes. Unlike conventional GP representation where each individual is represented by a single tree, each individual in multitree GP (MGP) [6] is represented by a *set* of trees as depicted in Fig. 2. Each individual in $\text{MGPD}^{\text{ri}}_{t,w}$ comprises a set of predefined number of trees (t), where the leaf nodes are drawn from the *terminal* set and the internal nodes are drawn from the *function* set. Similar to GP-criptor$^{\text{ri}}$ [1], the terminal set comprises four first-order statistic node types $min(\cdot)$, $max(\cdot)$, $mean(\cdot)$, and $stdev(\cdot)$. The aim of each of these node types is to perform feature extraction as they aggregate a set of pixels and return a single value. The function set consists of the arithmetic operators that are often used in GP such as $+$, $-$, $\times$, and protected / (returns 0 if the denominator is zero). One of the main differences between $\text{MGPD}^{\text{ri}}_{t,w}$ and the baseline GP-criptor$^{\text{ri}}$ method is that the *code* node type has been omitted, which represents the root node of an individual evolved by GP-criptor$^{\text{ri}}$. Having the *code* node type requires the use of STGP in order to define restrictions on the inputs and outputs of the different node types. Moreover, special care is required to ensure the closure property when applying the mutation and crossover operators. The *code* node converts the output of its children to a binary code by using the 0 value as a threshold. As this node is not used in $\text{MGPD}^{\text{ri}}_{t,w}$, the system applies the same rule, i.e., uses the 0 value as a threshold, on the output of the root node of each tree of the individual in order to generate a binary code from the outputs of those trees. More details on this operation are provided in Sect. 3.4.

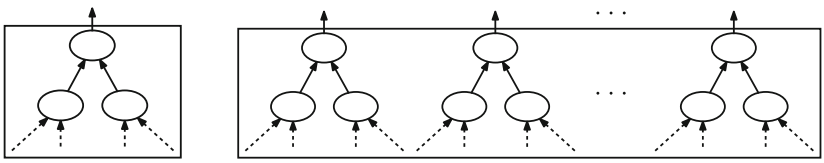

Fig. 2. GP individual representations (a) single tree; and (b) multitree.

3.3 Fitness Function

The fitness of an individual often reflects its ability to tackle the user-defined problem. Hence, the design of the fitness function is highly dependent on some factors such as the problem at hand (e.g. classification or regression) and the restrictions (e.g. number of training examples). For classification tasks, the accuracy measure, or its variations, is often used as a fitness function. However, relying on the accuracy is inappropriate in some situations such as when there are only a few training instances [1], or the dataset is highly imbalanced [2], i.e., the instances of one class are outnumbered by the instances in the other classes. Similar to GP-criptor$^{\text{ri}}$, the fitness function in $\text{MGPD}^{\text{ri}}_{t,w}$ is designed to consider the

distance between instances from different classes as well as the distance between instances of the same class, which is defined as

$$fitness = \alpha \times D_{\mathrm{w}} + (1 - \alpha) \times (1 - D_{\mathrm{b}}) \tag{5}$$

where α is a scale factor $\in [0, 1]$, and D_{w} and D_{b} are, respectively, the within-class and between-class distance components.

Unlike GP-criptor$^{\mathrm{ri}}$, MGPD$^{\mathrm{ri}}_{t,w}$ does not measure the average distance between each instance and *all* instances from the same and other classes; instead, the D_{w} component measures the average distance between each instance and only the *farthest* (most dissimilar) instance from the same class and calculated using Eq. (6); whilst D_{b} measures the average distance between each instance and only the *closest* (most similar) instance from all other classes and is calculated using Eq. (7). This design was motivated by the concepts of margins in support vector machines.

$$D_{\mathrm{w}} = \frac{1}{z} \sum_{\mathbf{u} \in \mathbf{R}} \sum_{\boldsymbol{u} \in \mathbf{u}} \max_{\boldsymbol{v}} \chi^2 (\boldsymbol{u}, \boldsymbol{v}), \quad \{\boldsymbol{v} \in \mathbf{u}, \boldsymbol{u} \neq \boldsymbol{v}\} \tag{6}$$

$$D_{\mathrm{b}} = \frac{1}{z\,(c-1)} \sum_{\mathbf{u} \in \mathbf{R}} \sum_{\mathbf{v} \in \mathbf{R} \setminus \mathbf{u}} \min_{\boldsymbol{u}, \boldsymbol{v}} \chi^2 (\boldsymbol{u}, \boldsymbol{v}), \quad \{\boldsymbol{u} \in \mathbf{u}, \boldsymbol{v} \in \mathbf{v}\} \tag{7}$$

Here a bold letter, e.g., $\mathbf{u}$ and $\mathbf{v}$, is used to indicate the set of all instances belonging to one class.

In MGPD$^{\mathrm{ri}}_{t,w}$, the aim is to minimise the fitness value, i.e., the smaller the fitness value the better the individual. Hence, the system will try to minimise D_{w} and maximise D_{b}. It is worth noting that χ^2 returns a value between 0 and 1 (inclusive), and subsequently the values for D_{w} and D_{b} are ranging between 0 and 1. Ideally, the system will evolve an individual that has a fitness value equals to 0 (i.e. 0 within-class average distance and 1 between-class average distance); whereas an individual with a fitness value of 1 (i.e. 1 within-class and 0 between-class average distances) is considered the worst case scenario.

3.4 Feature Vector Extraction

Each individual in MGPD$^{\mathrm{ri}}_{t,w}$ is an image descriptor that operates directly on the raw pixel values of an image and generates a feature vector. The individual scans the image being evaluated in a pixel-by-pixel manner from left to right and from top to bottom using a window of size w (i.e. $w \times w$ pixels). At each pixel (window position), the system calculates the terminals required by invoking the *min*, *max*, *mean* and *stdev* functions on the pixel values of the current window. The calculated terminals are then fed into the trees of the individual and recursively each tree will be evaluated from the leaf nodes to the root node. The value obtained from each root node will be substituted by 0 if it is negative and 1 otherwise. This will form a binary code that comprises t bits, where t is the number of trees. The binary code is then converted to decimal and the corresponding bin of the histogram is incremented as presented in Fig. 3.

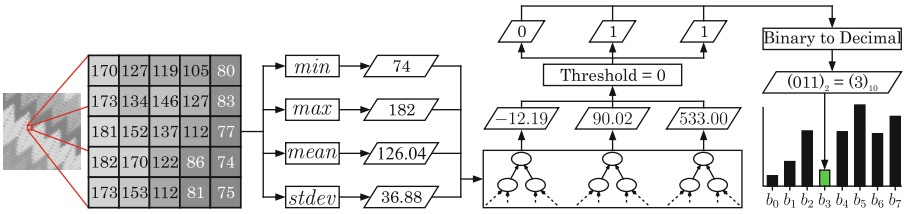

Fig. 3. An example demonstrates the steps to extract the feature vector from an image.

4 Experiment Design

To assess the performance of $\text{MGPD}^{\text{ri}}_{t,w}$, experiments are conducted using seven datasets for texture image classification. This section provides details regarding the benchmark datasets, methods for comparison, and parameter settings.

4.1 Benchmark Datasets

The proposed method is designed to handle grey-scale images, where the intensity of each pixel is ranging between 0 (black) and 255 (white). Hence, seven widely-used datasets for texture classification are selected in this study.

The first dataset in this study (BrNoRo) is formed from the widely used *Brodatz Texture*[1] [4] dataset. Originally, the Brodatz Texture dataset comprises 112 classes each of which consists of a single image of size 640×640 pixels. The single instance of 20 randomly selected classes is divided into 84 non-overlapping tiles each of which with size 64×64 pixels. The second dataset (BrWiRo) is formed by rotating the instances of BrNoRo around the centre at successive 30° angles, i.e., $\{0°, 30°, \ldots, 330°\}$. Figure 4 presents samples from BrNoRo.

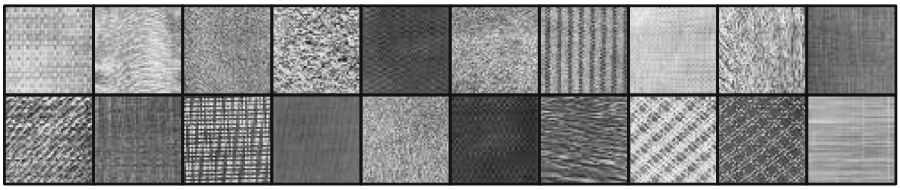

Fig. 4. Samples from the BrNoRo dataset.

The *Outex Texture Classification*[2] [17] test suites consist of 16 texture classification benchmark datasets that vary in illumination, rotation, spatial scale and colour. The third (OutexTC00) and fourth (OutexTC10) datasets in this study are formed using the instances of Outex_TC_00000 and its rotated version

[1] Available at: http://multibandtexture.recherche.usherbrooke.ca.
[2] Available at: http://www.outex.oulu.fi/index.php?page=classification.

Outex_TC_00010, respectively. Each of these two datasets consist of 24 classes for the same type of texture materials; however, the former dataset is rotation-free and the latter dataset comprises instances that fall into 9 rotation angles: 0°, 5°, 10°, 15°, 30°, 45°, 60°, 75° and 90°. Figure 5 shows examples from OutexTC00.

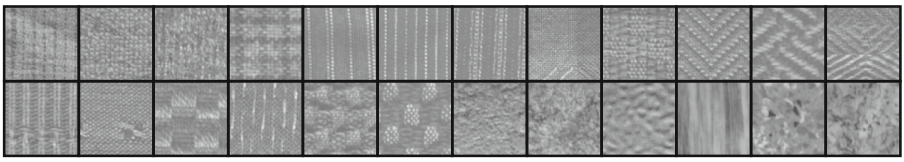

Fig. 5. Samples from the OutexTC00 dataset.

The fifth dataset in this study (KySinHw) is formed using the instances of *Kylberg Sintorn Rotation*[3] [13], which consists of 25 classes each of which comprises instances that were rotated at successive 40° angles, i.e., $\{0°, 40°, \ldots, 320°\}$. Figure 6 shows examples from this dataset.

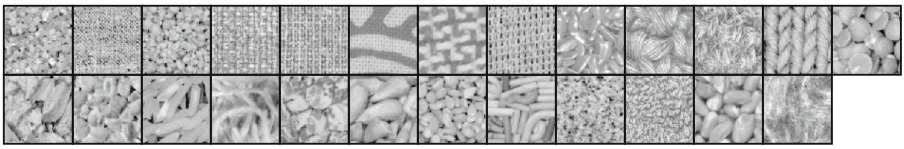

Fig. 6. Samples from the KySinHw dataset.

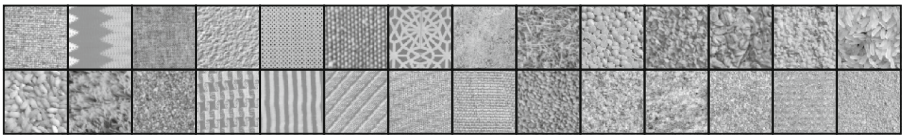

Fig. 7. Samples from the KyNoRo dataset.

The *Kylberg Texture*[4] [12] dataset comprises two groups: with rotation, and without rotation. The two groups consist of the same number of classes (28) and for the same type of materials as depicted in Fig. 7. The main difference between the content of these two groups is that the instances of each class in the former (without rotation) are all captured under the same rotation angle; whilst the instances of each class in the latter group (with rotation) are rotated in

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

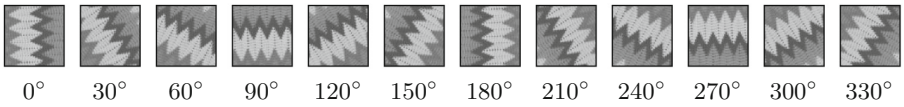

Fig. 8. A sample from the KyWiRo dataset presented in 12 rotation angles.

12° at successive 30° angles, i.e., $\{0°, 30°, \ldots, 330°\}$. Figure 8 shows an example taken from the with rotation group rotated in 12 angles. The sixth (KyNoRo) and seventh (KyWiRo) datasets in this study are, respectively, formed using the instances of the without and with rotation groups.

4.2 Methods for Comparison

The $\text{MGPD}^{\text{ri}}_{t,w}$ method aims at evolving dense image descriptors, therefore, six the-state-of-the-art dense hand-crafted image descriptors are used in this study in addition to the baseline method (GP-criptor$^{\text{ri}}$). The methods are uniform local binary pattern ($\text{LBP}^{\text{u2}}_{p,r}$) [18], uniform and rotation-invariant LBP ($\text{LBP}^{\text{u2ri}}_{p,r}$) [18], completed LBP ($\text{CLBP}_{p,r}$) [9], local binary count ($\text{LBC}_{p,r}$) and completed LBC ($\text{CLBC}_{p,r}$) [22], and dominant rotation LBP ($\text{DRLBP}_{p,r}$) [15].

4.3 Parameter Settings

Both $\text{MGPD}^{\text{ri}}_{t,w}$ and GP-criptor$^{\text{ri}}$ are evolutionary-based methods. The evolutionary parameters for both methods were kept identical as summarised in Table 1. Moreover, these methods comprise other non-evolutionary parameters. The window size (w) and number of trees (t), which is also the number of children of *code* in GP-criptor$^{\text{ri}}$, are experimentally set to 5×5 pixels and 9, respectively. The value of α in the fitness function of $\text{MGPD}^{\text{ri}}_{t,w}$ ranges between 0 and 1, which specifies the importance of the within-class and between-class distance components. Hence, 11 different values with a step of size 0.1 are used as shown in Fig. 9. As depicted in Fig. 9, the value of $\alpha \in [0.0, 0.6]$ gives good performance; therefore, the scale factor has been set to 0.1 in this study.

Table 1. The GP parameters

Parameter	Value	Parameter	Value
Generations	50	Minimum tree depth	2
Population size	300	Maximum tree depth	10
Crossover rate	80%	Mutation rate	20%
Selection type	Tournament	Tournament size	7
Elitism	Keep the best 10 individuals	Initial population	Ramped half-and-half

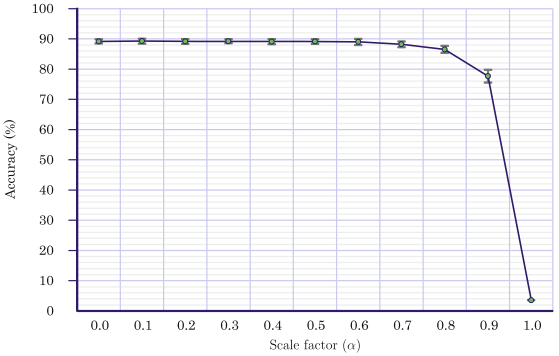

Fig. 9. The sensitivity of the scale factor (α) on KyNoRo.

Parameters of the benchmark methods have been investigated in [1], and therefore, they have been set at those values were observed to give the best performance. For $\text{LBP}^{\text{u2ri}}_{p,r}$, $\text{CLBP}_{p,r}$, $\text{LBC}_{p,r}$ and $\text{CLBC}_{p,r}$ methods, the radius (r) and number of neighbouring pixels (p) parameters have been, respectively, set to 3 and 24, i.e., $\text{LBP}^{\text{u2ri}}_{24,3}$, $\text{CLBP}_{24,3}$, $\text{LBC}_{24,3}$ and $\text{CLBC}_{24,3}$; whereas $\text{LBP}^{\text{u2}}_{p,r}$ and $\text{DRLBP}_{p,r}$ are set to $p = 8$ and $r = 1$, i.e., $\text{LBP}^{\text{u2}}_{8,1}$ and $\text{DRLBP}_{8,1}$.

4.4 Experiments

The main role of an image descriptor is to generate the feature vector for an image. The classification accuracy is widely adopted to measure the goodness of a descriptor [18]. Hence, the k-Nearest Neighbours classifier with $k = 1$ (1-NN) is used in this study. Apart from GP-criptor$^{\text{ri}}$ and $\text{MGPD}^{\text{ri}}_{t,w}$, all other methods are deterministic that require only a single run. The run for each of the stochastic methods, i.e., GP-criptor$^{\text{ri}}$ and $\text{MGPD}^{\text{ri}}_{t,w}$, is repeated independently 30 times using different seed values and the average performance is reported. The training set ($\mathbf{S_{tr}}$) is formed by randomly selecting two instances from each class (Sect. 3.1); therefore, the same process is further repeated 10 times and the mean and standard deviation are reported. In total, there are 4620 runs (= [(30 (runs) × 2 (methods)) + (1 (run) × 6 (methods))] × 10 (repeats) × 7 (datasets)).

5 Results and Discussions

The results of the eight methods on the seven datasets are presented in Fig. 10. Each block in this figure groups the results of a single dataset. To test the significance of the obtained results, Mann-Whitney-Wilcoxon Test is used with 0.05 significance level. The symbols "+", "−" and "=" are used in Fig. 10 to indicate that the proposed method is significantly better, significantly worse, and not significant, respectively, compared to the corresponding method.

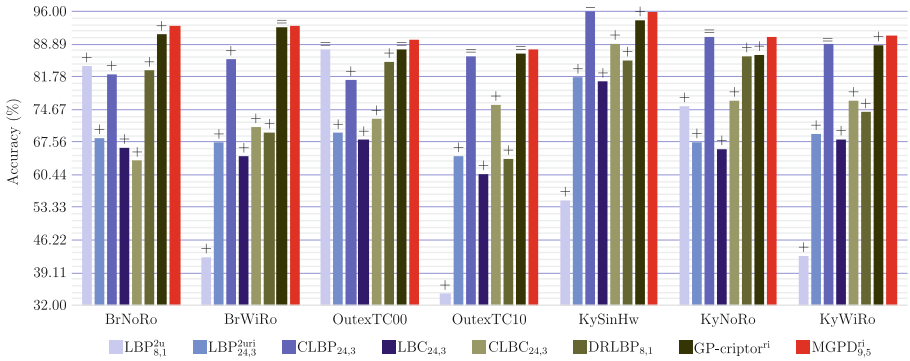

Fig. 10. The average accuracy (%) of eight image descriptors on seven texture datasets.

On the first dataset (BrNoRo), the proposed method has significantly outperformed all the other image descriptors, and achieved on average 93.00% accuracy. The minimum difference between $\text{MGPD}^{\text{ri}}_{9,5}$ and the other methods ranges between 2.08% (GP-criptor$^{\text{ri}}$) and 29.38% ($\text{CLBP}_{24,3}$).

The results on the second dataset (BrWiRo) show that $\text{MGPD}^{\text{ri}}_{9,5}$ has achieved the best performance (92.94%). Apart from GP-criptor$^{\text{ri}}$, $\text{MGPD}^{\text{ri}}_{9,5}$ has significantly outperformed the other methods.

The results on the third dataset (OutexTC00) show that $\text{MGPD}^{\text{ri}}_{9,5}$ has achieved on average 89.75% accuracy, which is significantly better than the performance of the competitor methods, apart from $\text{LBP}^{\text{u2}}_{8,1}$ and GP-criptor$^{\text{ri}}$.

The results on the fourth dataset (OutexTC10) show similar pattern compared to that of the rotation-free dataset (OutexTC00) with on average performance of 87.82%, where $\text{MGPD}^{\text{ri}}_{9,5}$ has significantly outperformed five methods and show comparable, yet better, result to $\text{CLBP}_{24,3}$ and GP-criptor$^{\text{ri}}$.

The proposed method has achieved the second best performance (95.88%) on the fifth dataset (KySinHw), which is significantly outperformed the other methods apart from $\text{CLBP}_{24,3}$ (97.31%).

The results on the sixth (KyNoRo) and seventh (KyWiRo) datasets show that $\text{MGPD}^{\text{ri}}_{9,5}$ has significantly outperformed the competitor methods apart from $\text{CLBP}_{24,3}$. The proposed method has achieved on average 90.53% on KyNoRo and 90.91% on KyWiRo.

6 Conclusions

This paper has successfully utilised multitree GP to automatically evolve rotation-invariant image descriptors. Relying on the between-class and within-class distances, the proposed method uses only two instances per class to evolve a descriptor. The results of the experiments on seven texture datasets show that the proposed method has significantly outperformed, or achieved comparable

performance to, six hand-crafted state-of-the-art methods and the baseline method (GP-criptor^{ri}).

In the future, we would like to investigate the proposed method for non-texture datasets. We also would like to further investigate the possibility of completely or partially transfer the evolved descriptors on one dataset to a similar domain (texture) or different, but related, domain (non-texture).

References

1. Al-Sahaf, H., Al-Sahaf, A., Xue, B., Johnston, M., Zhang, M.: Automatically evolving rotation-invariant texture image descriptors by genetic programming. IEEE Trans. Evol. Comput. **21**(1), 83–101 (2016)
2. Bhowan, U., Johnston, M., Zhang, M., Yao, X.: Reusing genetic programming for ensemble selection in classification of unbalanced data. IEEE Trans. Evol. Comput. **18**(6), 893–908 (2014)
3. Boric, N., Estevez, P.A.: Genetic programming-based clustering using an information theoretic fitness measure. In: Proceedings of 2007 IEEE Congress on Evolutionary Computation, pp. 31–38. IEEE (2007)
4. Brodatz, P.: Textures: A Photographic Album for Artists and Designers. Dover Publications, Mineola (1999)
5. Cha, S.-H.: Comprehensive survey on distance/similarity measures between probability density functions. Int. J. Math. Models Methods Appl. Sci. **1**(4), 300–307 (2007)
6. Cordella, L.P., de Stefano, C., Fontanella, F., Marcelli, A.: Genetic programming for generating prototypes in classification problems. In: Proceedings of 2005 IEEE Congress on Evolutionary Computation, pp. 1149–1155. IEEE (2005)
7. Ebner, M., Zell, A.: Evolving a task specific image operator. In: Poli, R., Voigt, H.-M., Cagnoni, S., Corne, D., Smith, G.D., Fogarty, T.C. (eds.) EvoWorkshops 1999. LNCS, vol. 1596, pp. 74–89. Springer, Heidelberg (1999). doi:10.1007/10704703_6
8. Fu, W., Johnston, M., Zhang, M.: Distribution-based invariant feature construction using genetic programming for edge detection. Soft Comput. **19**(8), 2371–2389 (2015)
9. Guo, Z., Zhang, L., Zhang, D.: A completed modeling of local binary pattern operator for texture classification. IEEE Trans. Image Process. **19**(6), 1657–1663 (2010)
10. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
11. Krig, S.: Computer Vision Metrics: Survey, Taxonomy, and Analysis, 1st edn. Apress, New York (2014)
12. Kylberg, G.: The Kylberg texture dataset v. 1.0. External report (Blue series) 35, Centre for Image Analysis, Swedish University of Agricultural Sciences and Uppsala University, Uppsala, Sweden (2011)
13. Kylberg, G.: Automatic virus identification using TEM: image segmentation and texture analysis. Ph.D. thesis, Division of Visual Information and Interaction, Uppsala University, Uppsala, Sweden (2014)
14. Lee, J.-H., Ahn, C.W., An, J.: An approach to self-assembling swarm robots using multitree genetic programming. Sci. World J. **2013**, 1–10 (2013)
15. Mehta, R., Egiazarian, K.: Dominant rotated local binary patterns (DRLBP) for texture classification. Pattern Recogn. Lett. **71**(1), 16–22 (2016)

16. Montana, D.J.: Strongly typed genetic programming. Evol. Comput. **3**(2), 199–230 (1995)
17. Ojala, T., Mäenpää, T., Pietikäinen, M., Viertola, J., Kyllonen, J., Huovinen, S.: Outex - new framework for empirical evaluation of texture analysis algorithms. In: Proceedings of 16th International Conference on Pattern Recognition, vol. 1, pp. 701–706. IEEE (2002)
18. Ojala, T., Pietikäinen, M., Mäenpää, T.: Gray scale and rotation invariant texture classification with local binary patterns. In: Vernon, D. (ed.) ECCV 2000. LNCS, vol. 1842, pp. 404–420. Springer, Heidelberg (2000). doi:10.1007/3-540-45054-8_27
19. Olague, G., Trujillo, L.: A genetic programming approach to the design of interest point operators. In: Melin, P., Kacprzyk, J., Pedrycz, W. (eds.) Bio-inspired Hybrid Intelligent Systems for Image Analysis and Pattern Recognition. SCI, vol. 256, pp. 49–65. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04516-5_3
20. Poli, R., Langdon, W.B., McPhee, N.F.: A Field Guide to Genetic Programming (2008). Published via http://lulu.com. (With contributions by J.R. Koza)
21. Willis, A., Sui, Y.: An algebraic model for fast corner detection. In: Proceedings of 12th IEEE International Conference on Computer Vision, pp. 2296–2302. IEEE (2009)
22. Zhao, Y., Huang, D.-S., Jia, W.: Completed local binary count for rotation invariant texture classification. IEEE Trans. Image Process. **21**(10), 4492–4497 (2012)