Genetic Programming for Region Detection, Feature Extraction, Feature Construction and Classification in Image Data

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

School of Engineering and Computer Science,
Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand
{Andrew.Lensen,Harith.Al-Sahaf,Mengjie.Zhang,Bing.Xue}@ecs.vuw.ac.nz

Abstract. Image analysis is a key area in the computer vision domain that has many applications. Genetic Programming (GP) has been successfully applied to this area extensively, with promising results. High-level features extracted from methods such as Speeded Up Robust Features (SURF) and Histogram of Oriented Gradients (HoG) are commonly used for object detection with machine learning techniques. However, GP techniques are not often used with these methods, despite being applied extensively to image analysis problems. Combining the training process of GP with the powerful features extracted by SURF or HoG has the potential to improve the performance by generating high-level, domain-tailored features. This paper proposes a new GP method that automatically detects different regions of an image, extracts HoG features from those regions, and simultaneously evolves a classifier for image classification. By extending an existing GP region selection approach to incorporate the HoG algorithm, we present a novel way of using high-level features with GP for image classification. The ability of GP to explore a large search space in an efficient manner allows all stages of the new method to be optimised simultaneously, unlike in existing approaches. The new approach is applied across a range of datasets, with promising results when compared to a variety of well-known machine learning techniques. Some high-performing GP individuals are analysed to give insight into how GP can effectively be used with high-level features for image classification.

Keywords: Genetic programming · Image classification · Feature extraction · Feature construction

1 Introduction

A common technique used in computer vision is the creation of features which provide a representation of an image that is of a higher level than that of the raw image pixels [12]. Many image classification approaches extract features from an image using a feature extraction algorithm, and then use these features as inputs to a machine learning algorithm to perform classification. A wide range of

M. Heywood et al. (Eds.): EuroGP 2016, LNCS 9594, pp. 51–67, 2016.
DOI: 10.1007/978-3-319-30668-1_4

algorithms for feature extraction have been proposed [17]. One popular approach is the Histogram of Oriented Gradients (HoG) algorithm [8], which produces a histogram of the gradients within an image which can then be used as a feature.

Genetic Programming (GP) has also been applied extensively to image analysis problems [21] since it was introduced in the 1990s. Techniques generally use GP to extract features from raw images by using pixel statistics [24,26], sliding window [22] or filter [3] approaches. GP is able to achieve success on these problems using its evolutionary learning process which allows it to automatically extract and construct high-level features tailored to the dataset it is trained on. This is in contrast to other algorithms such as HoG which do not have a learning process; these algorithms produce general, domain-independent features. The GP approaches tend to extract relatively simple features in comparison to the histograms produced by the HoG algorithm, which might limit their performance. Combining the training process of GP with the powerful features extracted by HoG may improve performance by generating high-level, domain-tailored features automatically. The literature contains many feature extraction methods; we use HoG in this work due to it being one of the most prevalent methods that is simple and efficient enough to implement as a GP function.

Another technique for improving feature quality is to only select regions of an image which are rich in useful features. A two-tier GP (2TGP) [2] method was proposed which automatically selects regions for feature extraction. Using this method in conjunction with more advanced GP feature construction functions would allow region selection and feature construction to be performed simultaneously to improve the image classification performance.

Goals. The goal of this paper is to develop a GP approach to automatically extract and construct high-level features for image classification. To achieve this, we propose a new GP-HoG approach which uses GP with functions based on the HoG method. These new functions are designed to produce more advanced features than the existing GP approaches. In this way, GP will be used for simultaneous region selection, feature extraction and image classification. We aim to achieve this through the following objectives: (1) developing new functions which are inspired by the HoG algorithm. These functions will allow GP to automatically produce high-level features which have the potential to increase classification performance; (2) combining these new functions with a region selection approach 2TGP to allow GP to perform region selection, feature extraction and classification in a single GP tree; (3) analysing the program trees of some good individuals to understand how they are able to generate useful features.

2 Background

Evolutionary Computation (EC) is a large field of artificial intelligence which contains algorithms inspired by biological evolutionary principles [6]. These algorithms are often applied to difficult problems, where the search space is very large. EC algorithms operate iteratively, refining the candidate solutions to a

problem in order to gradually improve solutions towards the optimal solution. Evolutionary Algorithms (EAs) are a field of EC algorithms which use Darwinian evolutionary principles to improve solutions by mimicking natural evolution [9].

Genetic Programming (GP) [13] is an EA which models solutions in the form of computer programs. The most common representation is a tree structure, where the root of the tree is the output of the genetic program and the leaves of the tree are inputs or constant values. Non-terminal nodes are functions in the program, which take some inputs (i.e. outputs of other nodes), and then produce an output based on a function applied to those inputs. Terminal nodes are the leaves of a tree.

Feature construction is the process of creating new, high-level features, often by combining multiple existing features [4,14]. Constructed features generally better describe an instance than a single existing feature, reducing the number of features required, which reduces the size of the search space a classifier must train on. GP has been applied extensively to feature construction tasks [10], due to its tree-structure which allows features to be combined using a range of functions to create new features. As GP generally produces a single output value from the root, techniques often use it to produce a single high-level feature.

Al-Sahaf *et al.* [5] proposed a GP approach to automatically construct an image descriptor that is then used to extract features for multi-class texture classification. Their experiments present the capability of the method to extract important features. The method has significantly outperformed the competitor methods on two texture data sets.

The HoG [8] technique produces a feature vector from an image based on the orientation of gradients within the image. The image is first split into a number of overlapping blocks. Each block produces a histogram of gradients of pixels within that block. For each pixel in a block, both the magnitude and the orientation of its gradient is recorded. The histogram of each block, then, contains bins for various orientations (one bin for a range of orientations), and the height of each bin is the sum of the magnitudes of the gradients falling within that bin. The histogram from each block is then normalised, and all histograms are then combined to give a final feature vector corresponding to the image as a whole. This kind of feature vector has been used for a variety of image analysis problems [8,27].

2.1 Related Work

This subsection briefly surveys typical related work which uses GP to extract and construct features for image classification. The limitations of these works are discussed, showing the motivation behind our proposed GP-HoG approach.

The 2TGP approach [2] used GP to select good regions of an image, extract features from the regions (as simple statistics based on the pixels in the region), and to perform classification. The approach was tested on a variety of datasets which varied in difficulty, with good results across different image domains. The solutions produced were also easy to understand. For example, on a face dataset,

solutions were produced using regions which humans would also use for classification, such as the nose, mouth and eyes. On a pedestrian dataset, regions were selected which captured areas, where a standing pedestrian would be expected to appear. By selecting regions, this approach was able to improve classification accuracy. Using more advanced features, i.e., beyond simple pixel statistics, in combination with region selection has the potential to improve the performance even further. This is the direction we take in this study.

In [23], the root of the tree was used as a constructed feature by a Support Vector Machine (SVM) for image classification. This approach created GP trees using a large range of functions which directly considered the pixel values of the given images. By using a multi-objective approach which tried to minimise tree size while maximising classification accuracy, the authors were able to reduce over-fitting and achieved a high classification accuracy. The function set used a range of filtering functions including Gaussian, Laplacian, and Gabor filters, as well as simpler pixel-by-pixel arithmetic operations. While these filtering functions are more advanced than simpler pixel statistic approaches, they are still relatively simpler than the HoG algorithm, as they apply a small filter to each pixel instead of using a more sophisticated histogram technique.

Perez and Olague proposed a GP technique (RDGP) [20] using a function set and a terminal set, which was designed to emulate the Scale-Invariant Feature Transform (SIFT) [15], another widely-used feature extraction algorithm. A range of functions and terminals were used, including arithmetic operators, image derivatives and Gaussian filters. The authors argued that their method would allow GP to automatically synthesise SIFT-like programs by automatically extracting high-level features for object recognition. They claimed that their approach allowed features to be automatically tailored towards the problem being trained on, as the GP programs would be optimised by the evolutionary process. The RDGP approach was shown to produce better features than the standard SIFT approach, with an overall decrease in error in object detection. While this approach performed feature extraction and construction, it did not use GP for classification. As their design broke the SIFT algorithm into its composite parts as GP functions, the evolutionary process must learn to re-construct and optimise these composite parts in order to produce useful features. This may reduce the performance of the method; the approach we propose attempts to mimic HoG within a single function, so that the evolutionary process can instead focus on constructing high-level features and simultaneously evolve a good classifier.

3 The Proposed Method

This section details the proposed method (named GP-HoG) including the program representation, and the fitness function.

3.1 GP Program Representation

The proposed method uses a combination of existing terminals and functions from 2TGP and novel terminals and functions inspired by HoG. In this study, a

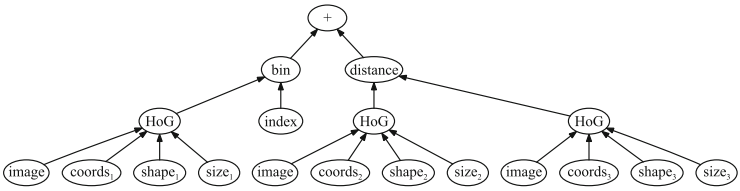


Fig. 1. An example shows an individual structure of the GP-HoG method.

new tree-based program structure is proposed, as presented in Fig. 1. To introduce restrictions on the inputs and outputs of the different nodes in an evolved program, *strongly-typed GP* [18] is used. The full terminal and function sets are listed in Tables 1 and 2 respectively. An individual's tree can be, virtually, divided into three layers. The bottom layer, which includes the HoG and terminal nodes, represents the feature extraction part. The middle layer, which consists of a mix of the *bin* and *distance* nodes, represents the feature construction part. The top layer (including the root node) that is made up of a chain of simple arithmetic operators represents the classification part. As Strongly-Typed GP is used, all three layers appear in every valid program. The feature construction layer represents a main difference between GP-HoG and 2TGP, which has only feature extraction (as pixel statistics across a region) and classification layers. The new feature construction layer aims to further reduce the search space by constructing high-level features from the extracted features from the previous layer, with the expectation of evolving meaningful classifiers and further improving the performance.

The majority of the terminal set is based on the 2TGP method, as the terminals provide the parameters used in the region selection process. The *image* node is the instance, i.e., image being evaluated represented as a 2D array of integer values each of which represents the intensity of a pixel in the image. *rand* is a random value drawn from the half-closed interval $[0, 1)$. The *shape* node defines the shape of a region that can be `rectangle`, `circle`, `row` and `column`. The *coords* node is a pair of (x, y) values that define the location of a region in an image where (x, y) is the centre pixel for the circular shape and the top-left corner pixel for all other shapes. The minimum width and height of all images in the dataset are, respectively, denoted as min_{Width} and min_{Height}. T The *size* node specifies the size of a region. The value of a *size* node is defined to be 1 in `row` and `column` regions. The *size* node is not used in the case of a rectangle region and is replaced by (w, h) which are the *width* and *height* of the region. For a circular region, the *size* node gives the diameter. Restricting the dimensions of regions by using min_{Width} and min_{Height} encourages regions to be created which are valid across the majority of images in a dataset, which improved training and test performance. In order to return a specific value of a histogram, the *index* node is introduced that takes a value between 0 and 7 (inclusive) as each histogram consists of 8 bins (as detailed in the next subsection).

Table 1. The terminal set

Name	Output	Details
image	image	The image being evaluated represented as a 2D array of pixel values
rand	double	Random double in the half-closed interval $[0, 1)$
coords	coords	Provides the location as a region as (x, y). x is the horizontal location randomly generated in $[0, min_{\text{Width}}]$ and y is the vertical location in $[0, min_{\text{Height}}]$
size	size	Random integer value in $[3, \min(min_{\text{Width}}, min_{\text{Height}})]$
index	index	Random integer value in $[0, 7]$, which represents the bin index of a histogram
shape	shape	One of rectangle, circle, row, and column. Rectangle has x in $[0, min_{\text{Width}}]$ and y in $[0, min_{\text{Height}}]$

Table 2. The function set

Name	Input	Output	Details
$+, -, \times, /$	double, double	double	Arithmetic operators
bin	histogram, index	double	Returns the value of the specified index
HoG	image, coords, shape, size	histogram	Performs the HoG algorithm on a region
distance	histogram, histogram	double	Returns the distance between two histograms

The function set comprises of the four arithmetic operators $+$, $-$, $\times$ and protected $/$, *distance*, *bin*, and *HoG* nodes. The arithmetic operators in the function set have their corresponding regular meaning and allow GP to utilise multiple extracted features for classification. The *HoG*, *distance* and *bin* functions and the *index* terminal are used for feature extraction. The design of these functions is discussed in the next subsection.

The most important new function is the HoG function, which is inspired by the HoG algorithm [8]. The HoG function takes the *image*, *coords*, *shape*, and *size* as inputs, and outputs a histogram which represents the distribution of gradients within a region of the image. The standard approach of using a histogram with 8 bins [8] is adopted here, where each bin corresponds to 45° of rotation. By using the provided terminal nodes, a GP tree can construct a histogram across a region of varying shape and size; there are no pre-determined regions as in the normal HoG method. This allows the regions used to be tailored to the dataset.

3.2 Outline of the HoG Function

The region of the image is selected based on the inputs to the *HoG* function. This is done by taking a region of an image defined by the *shape* (how the region is shaped), *coords* (the position of the region) and *size* (size of the region) inputs. Then, the steps presented in Algorithm 1 are applied.

Our HoG approach differs from the standard HoG algorithm [8] in a few ways in order to allow it to be expressed sensibly as a function for GP. The biggest

difference is that the *HoG* function is applied only to a single region, given by the function arguments. Normally, the histograms of multiple overlapping blocks across an image are combined to give a more versatile feature vector. As multiple *HoG* functions can be incorporated in a single GP tree, it is not necessary to use multiple blocks to try and analyse the whole image; this will be done automatically as part of the evolutionary process if it gives good performance. As only a single histogram is produced from one run of the proposed GP-HoG algorithm, the normalisation process is only applied to a single histogram, rather than across several as in the original design. This approach also allows for a range of block (i.e. region) sizes and shapes; normally the design of blocks are fixed, such as using an 8×8 square. It is expected that the evolutionary process will be able to automatically find the best block sizes and shapes as individuals with the best block designs, i.e., more representative features, will be rewarded with a better fitness value. The crossover operator allows useful block designs to be exchanged between individuals.

As a number of variations to the original HoG algorithm have been made, we consider Algorithm 1 to be inspired by the HoG algorithm, rather than being a strict implementation of it. As Algorithm 1 outputs a feature vector (histogram) of eight bins, GP can not directly use this vector for classification; therefore, two additional functions were designed which construct high-level features from a histogram. The first function is *distance*, which finds the Euclidean distance between two histograms. This produces a double value which gives a measure of how dissimilar two histograms are. This can be used to compare different regions of an image in order to identify the image's class. For example, on the UIUC dataset (Sect. 4), the regions corresponding to a car's front and back wheels will produce similar histograms. These same regions on a background image are more likely to give different histograms. Hence, the distance between histograms can be used as a feature for classification. The second function is *bin*, which returns the value of a given bin index of a histogram. This function allows GP to select important orientations which have different magnitudes depending on the image class. For example, on the Jaffe dataset, the edges of the mouth have different gradients for the subjects being happy or surprised. The magnitude of a given bin can be used as a feature for distinguishing two classes.

3.3 The Fitness Function

The evolutionary process measures a program's goodness using the fitness function. In this work, the accuracy of a program is used as the fitness value to reflect its ability to discriminate between instances of different classes. The accuracy is the proportion of correctly classified instances to the total number of instances. Hence, an ideal program will have a fitness value of 1 and a fitness value of 0 represents the worst case scenario or performance.

Classification of an instance is performed by feeding it into an evolved GP tree. The *image* terminals of the GP tree are set to contain the image being classified, and then the tree is evaluated from bottom to top, producing a single real number as an output. A threshold 0 is then applied to this value.

Algorithm 1. The procedure used in the HoG function

1 Find the gradient for each pixel as $G_x = f(x+1, y) - f(x-1, y)$ and $G_y = f(x, y+1) - f(x, y-1)$ where $f(i, j)$ gives the pixel value at (i, j).
2 Find the magnitude at each pixel as $m = \sqrt{G_x^2 + G_y^2}$.
3 Find the orientation of each pixel as $\arctan\left(\frac{G_y}{G_x}\right)$. This is converted to degrees and mapped to be in range $[0°, 360°]$.
4 **for** each pixel **do**
5 Find the two bins of the histogram it lies between based on its orientation. The histogram is divided into 8 bins of size 45°, so each pixel with an orientation will have a lower bin and an upper bin. For example, a pixel with an orientation of 80° would have its lower bin as bin 2 (45°), and its upper bin as bin 3 (90°).
6 For each bin, find the distance between the bin's orientation and the pixel's orientation. In the previous example, an orientation of 80° puts that pixel at 35° distance from the lower bin, and 10° from the upper bin.
7 For each bin, calculate and add the weighted magnitude as the pixel's magnitude multiplied by how close it is to that bin. As the bin size is 45°, $m \times \frac{(45-35)}{45}$ is added to the lower bin. The upper bin would have $m \times \frac{(45-10)}{45}$ added to it, as the upper bin's orientation is closer to that of the pixel.
8 **end for**
9 Normalise the histogram by expressing the value of each bin as a fraction of the sum across all bins.

A negative value gives a negative classification, and a non-negative value gives a positive classification. A tree may contain regions that partially fall outside the dimensions of the image. Any such regions are cropped, so only the pixels within the image bounds are used in the computation of the histogram in the HoG function.

4 Experiment Design

This section details the datasets, parameter settings, and methods for comparison used in this study.

4.1 Datasets

Three datasets were used to assess the performance of the proposed method. These datasets are for different applications and vary in difficulty. However, each of the datasets is made up of greyscale images and is set for binary classification.

In computer vision, the *Columbia Object Image Library*[1] (COIL-20) [19] dataset is widely used. Two classes of the COIL-20 dataset are used to form the first dataset in this study. Originally, the COIL-20 dataset comprises of 20 classes that each represents a different toy object, e.g., cars, rubber ducks, and boxes. A turntable is used in a scene with a black background to prepare those instances.

[1] Available at: http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php.

For each object, 72 images are provided by taking a snapshot every 5° where the object is rotated through 360°. Then those images are cropped to be 128×128 pixels each where the object is centred in the images. Furthermore, those images were normalised by adjusting the brightest pixel to be 255 and scaling the other pixel values accordingly. In this study, only the *cars* and *rubber ducks* classes (Fig. 2(a)) are used as the focus is on performing binary classification.

Meanwhile, the second dataset in this study is formed using the *Japanese Female Facial Expression*[2] (Jaffe) [16] dataset. This dataset is broadly used in the literature for the task of identifying different facial expressions. In total, this dataset consists of 213 images provided by ten Japanese female subjects, which is divided into seven groups: neutral, surprised, angry, sad, happy, disgust, and fear. Each subject provides several images for each facial expression. Following Cheng *et al.* [7], and in order to prevent the classifiers from training on irrelevant features, the images of this dataset were manually cropped in order to remove most of the subject's hair, and the image background leaving only the face. The size of those instances after cropping ranges between 164 and 207 pixels in height, and between 121 and 143 pixels in width. The instances of the happy and surprised classes (Fig. 2(b)) are used in this study to form the second dataset.

To form the third dataset in this study, the *UIUC database for Car Detection*[3] (UIUC) dataset [1] is used. In total, the UIUC dataset consists of $1,050$ instances that fall into two classes: cars and background (Fig. 2(c)). The former comprises of 550 instances, whilst there are 500 instances in the latter. The car instances are captured from the same angle and distance (giving the same scale) that show the side view of the vehicle. Each instance in this dataset is 100×40 pixels.

(a) (b) (c)

Fig. 2. Samples of the (a) COIL-20, (b) Jaffe, and (c) UIUC cars datasets showing instances of the positive and negative classes in the top and bottom rows respectively.

4.2 Training and Test Sets

The k-fold cross-validation technique was used to evaluate the proposed method and all the baseline methods. The instances of the UIUC and COIL-20 datasets were randomly split into 10 folds. The Jaffe dataset contains only three images of each expression for each human subject, requiring a careful split to ensure that

[2] Available at: http://www.kasrl.org/jaffe.html.
[3] Available at: http://cogcomp.cs.illinois.edu/Data/Car/.

the test and training sets are both representative of the dataset, and therefore, this dataset was manually split into 3 folds. Each fold contained one happy and one surprised expression for each subject.

4.3 Baseline Methods

A number of baseline methods are used to compare the new approach to the existing methods in the literature. The Waikato Environment for Knowledge Analysis (WEKA) [11] implementations of the Support Vector Machine (SVM), Decision Trees (J48), Naïve Bayes (NB), Random Forest (RF), and Adaptive Boosting (ABM1) classifications methods are used [25]. As the proposed method is largely based on 2TGP, the 2TGP method is also used as a competitive method in this study. Each of these seven methods (including the proposed method) were evaluated using the k-fold cross-validation scheme described in Sect. 4.2. For each of the non-GP classifiers, an instance is evaluated by giving the classifier a list of concatenated SURF keypoints (as detailed in the next subsection). As a keypoint contains 64 values, there will be $p \times 64$ features provided, where p is the number of keypoints used. For example, if two keypoints called a and b were used, the list would be formatted in the form $[a_0, a_1, .., a_{63}, b_0, b_1, ..., b_{63}]$. As the WEKA implementation of each of these methods is deterministic, they are only run once for a given experiment run. The list of keypoints is ordered by the strength of each keypoint so that the methods are able to learn most effectively. The SURF feature extractor generates keypoints based on the location of the keypoint in the image, which means that a classifier may classify two instances of the same class differently depending on the distribution of keypoints throughout the images, even if the images are actually similar. For example, if two images were of the same person's face but in one the face was shifted 50 pixels to the right, the keypoint corresponding to a "nose feature" could appear in different locations in the keypoint list. By ordering keypoints by how strong they are, the classifier is more likely to classify similar instances to the same class as they will likely have similar strong keypoints at the same index in the list.

4.4 Generating SURF Keypoints

Both of the GP-based (2TGP and GP-HoG) methods are designed to operate directly on the raw pixel values, which is not the case for the other baseline methods. Therefore, SURF image descriptor is used to generate a list of keypoints that can be used as a high-level features by those classifiers. However, SURF generates varying numbers of keypoints based on the number of interest points an image has. For many classifiers, this presents a problem as a static number of features (fixed length feature vector) is expected. Algorithm 2 is developed to address this problem. This method relies on altering the Hessian threshold, which represents a main component of the SURF method, in order to determine the interest points. Binary search is used to adjust this threshold until a predefined number of keypoints are retrieved. A fixed number of keypoints allows more effective training as a solution can perform consistently across a dataset.

Algorithm 2. Selecting top-p keypoints

```
1  function selectKeypoints(image, L_bound, U_bound, p)   ▷ where L_bound and U_bound are the
2      threshold ← (U_bound − L_bound)/2 + L_bound            lower and upper bounds of the
3      keypoints ← SURF(threshold)                            Hessian threshold, respectively.
4      if |keypoints| = p then
5          return keypoints
6      else if |keypoints| > p then
7          L_bound ← threshold
8      else
9          U_bound ← threshold
10     end if
11     return selectKeypoints(image, L_bound, U_bound, p)
12 end function
```

4.5 Evolutionary Parameters

GP has a number of parameters which can be altered in order to optimise the evolutionary process for a given problem. The GP-HoG and the 2TGP methods were applied to the three datasets (Sect. 4.1). For 2TGP, the same parameters were used as in [2]; namely, 80 % crossover, 20 % mutation and top-10 elitism was used. The population size was 1,024, and the minimum and maximum tree depth were 2 and 10, respectively. On each dataset, the evolutionary process was independently executed 35 times using different seed values. Each execution was run for 50 generations or until perfect training performance was obtained. GP-HoG used 40 % mutation and 60 % crossover as it was found a higher mutation rate could produce better training performance by allowing a wider exploration of the search space. All other parameters were the same as for 2TGP.

5 Results and Discussion

This section compares the performance of the GP-HoG approach to the 2TGP approach and the five SURF baselines. It also discusses the increase in training time required to train GP using the GP-HoG approach compared to using 2TGP.

5.1 Compared to the 2TGP Approach

The results of the 2TGP and GP-HoG are shown in Table 3. Student's t-test with a 95 % confidence interval was used to evaluate the significance of the performance increase using GP-HoG. A "+" in Table 3 indicates that GP-HoG is significantly better than the 2TGP approach, whereas a "−" indicates it is significantly worse. The GP-HoG approach performs significantly better on the Jaffe and UIUC datasets (the two difficult datasets) while achieving slightly worse mean (but identical maximum) test performance on the COIL-20 dataset. On the most difficult dataset (Jaffe), GP-HoG achieves a 5 % and 11 % increase in mean and maximum test performance respectively over the 2TGP approach.

The average training time has increased notably using the new approach compared to 2TGP, with approximately 3× more computation required on the Jaffe and UIUC datasets, likely due to the larger amount of computation required by

the *HoG* function than in the aggregation functions used in the 2TGP approach. The arctan $(\cdot)$ function is the slowest part of the HoG algorithm (from empirical sampling), as it is somewhat expensive to compute even on a modern CPU, and is used once for every pixel in a region. On the UIUC dataset (which has the largest number of images), training never finishes before the maximum number of generations (the maximum training performance is 98 %), and so the training time is much longer than the other datasets. Even with the utilisation of multi-threading, the 600 hours of CPU time across all folds takes about a week of real time. It is important to note that while the increase in training time is a downside of the new approach, the time required to apply the best trained solution to new, unseen images is still minimal. Long training times are common when GP is used, but as long as the evolved programs are not overly complex, they are often quick enough to be used on unseen instances.

5.2 Compared to the Baselines

The results of the five non-GP methods on the three datasets using different numbers of SURF keypoints are presented in Table 4. The values of the last two blocks ($p = 20$ and $p = 50$) of the UIUC dataset are not available as SURF could not generate this many keypoints due to a lack of interest points in the images in this dataset. The GP-HoG approach has similar performance on COIL-20 and improved performance on the Jaffe and UIUC datasets compared to the non-GP baselines. This is unsurprising, as the GP-HoG approach is able to perform region selection and feature construction to give more advanced and dataset-specific features than the domain-independent features produced by SURF.

Table 3. The accuracy and average training time (H:M:S) of the 2TGP and GP-HoG methods on the three datasets.

	2TGP						GP-HoG					
	COIL-20		Jaffe		UIUC		COIL-20		Jaffe		UIUC	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Max	1.00	1.00	0.98	0.81	0.96	0.94	1.00	1.00	0.99	0.92	0.96	0.95
Mean	1.00	0.99	0.95	0.71	0.94	0.92	1.00	0.97^{-}	0.97^{+}	0.76^{+}	0.95^{+}	0.93^{+}
St.Dev	0.00	0.01	0.02	0.05	0.01	0.01	0.00	0.02	0.02	0.07	0.01	0.01
Training time	00:05:49		01:35:54		16:22:39		03:24:26		04:36:24		52:15:25	

6 Further Analysis

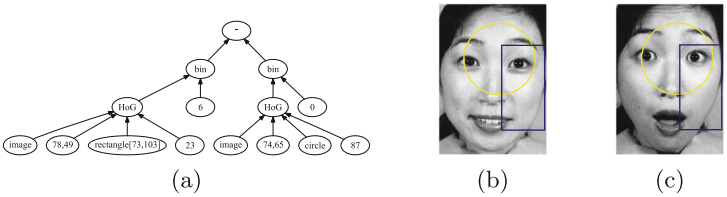
The GP-HoG approach produces programs which can be interpreted and understood by humans. This section analyses three high-performing evolved programs to understand how they can perform classification with high accuracy.

Table 4. The average accuracies of the non-GP baseline methods on the three datasets.

		$p = 5$		$p = 10$		$p = 20$		$p = 50$	
		Train	Test	Train	Test	Train	Test	Train	Test
COIL-20	SVM	1.00	0.99	1.00	0.99	1.00	0.99	1.00	1.00
	J48	0.99	0.86	0.99	0.86	0.99	0.86	0.99	0.85
	NB	0.99	0.97	0.99	0.96	1.00	0.91	1.00	0.94
	RF	1.00	0.99	1.00	0.99	1.00	1.00	1.00	1.00
	ABM1	1.00	0.96	1.00	0.96	1.00	0.93	1.00	0.93
Jaffe	SVM	1.00	0.63	1.00	0.72	1.00	0.74	1.00	0.82
	J48	0.98	0.70	0.98	0.56	0.98	0.59	0.98	0.79
	NB	0.89	0.72	0.92	0.72	0.98	0.75	1.00	0.69
	RF	1.00	0.77	1.00	0.76	1.00	0.77	1.00	0.71
	ABM1	1.00	0.70	1.00	0.77	1.00	0.67	1.00	0.71
UIUC	SVM	0.99	0.92	1.00	0.91	N/A	N/A	N/A	N/A
	J48	0.99	0.84	0.99	0.83	N/A	N/A	N/A	N/A
	NB	0.90	0.89	0.90	0.89	N/A	N/A	N/A	N/A
	RF	1.00	0.94	1.00	0.93	N/A	N/A	N/A	N/A
	ABM1	0.88	0.85	0.88	0.84	N/A	N/A	N/A	N/A

6.1 Example Program 1

An evolved program with high performance on the Jaffe dataset is shown in Fig. 3. This program is interesting to analyse, as it is very simple, consisting of two *HoG* operators, and a subtraction operator. The left side of the tree applies the *HoG* operator to a rectangular region corresponding to the right side of the face, including the eye, cheek, and part of the nose and lip areas. This region contains different features of the face in the happy and surprised expressions. When the subject is happy, the corner of the mouth is narrower than when surprised, producing a smaller gradient orientation. When they are surprised, the mouth is widened, creating a right-angle between the chin and where the subject's ear would be. This edge is at a larger angle than when the subject is happy, and hence a different histogram is produced. The *HoG* operator produces a histogram, and the value of the bin corresponding to the 270°–314° orientation range is selected by the *bin* function. On the right side of the tree, the *HoG* operator is applied to a circular region of the image, which corresponds to the upper nose, eyes, and eyebrow areas. The subject's nose appears much narrower when surprised, due to her mouth being open. There is also more of the nose included in the surprised image, which introduces an additional edge gradient. The eyes are also different when surprised; they appear wider and have more white showing. All of these differences change the histogram that is produced, allowing GP to extract features that distinguish these two expressions. The value of the bin corresponding to the 0°–44° range is chosen by the *bin* node on the right side of the tree. The root of the tree then outputs the difference between the values from the left and right sides of the tree. This program scores 98 % and 95 % on training and test sets respectively on the fold it was generated in.

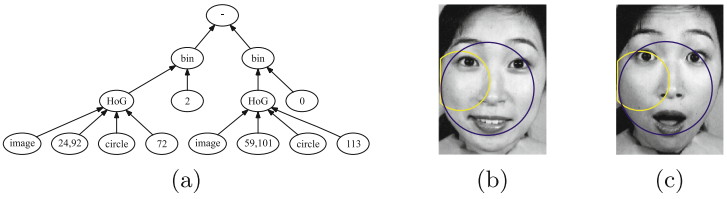


(a) (b) (c)

Fig. 3. Example program 1: 98 % training and 95 % test performance on Jaffe dataset (a) GP tree, (b) Happy (+ve) expression, and (c) Surprised (−ve) expression.

6.2 Example Program 2

Another program with very good performance (95 % training and 100 % test accuracy) on the Jaffe dataset is shown in Fig. 4. This program is similar to Program 1 in that it uses only the difference of two histogram values to classify images with high accuracy. The yellow circular region contains the left eye and cheek, and a small part of the nose. The left eye has a different appearance between the happy and surprised expressions; when surprised, a larger amount of the eye is visible. The left nostril is also open in the surprised expression, producing a large gradient around it which is included in the yellow circle. While the blue circle is more difficult to analyse as it covers a large part of the image, one important observation is that it appears to be bounded by the eyes and mouth. The mouth is much darker in the surprised expression, giving a smaller gradient than in the happy expression where there is a distinct gradient between the white teeth and darker lips. By selecting the important mouth, eye and nose regions, the feature selected from the blue circle are used by GP to distinguish between happy and surprised expressions.



(a) (b) (c)

Fig. 4. Example program 2: 95 % training and 100 % test performance on Jaffe dataset (a) GP tree, (b) Happy (+ve) expression, and (c) Surprised (−ve) expression.

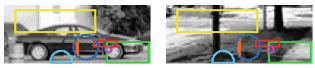
6.3 Example Program 3

Unlike the Jaffe and COIL-20 datasets, GP-HoG is unable to obtain perfect training performance on UIUC, and so large programs are produced by the evolutionary training process to maximise the training performance. The program in Fig. 5 (a) is one of the simpler programs that performs well on UIUC, with a

tree depth of eight. The regions used by this program are shown in Fig. 5 (b) and (c). While many regions are used, most tend to be small, identifying particular aspects of the image that are useful for classification. Several of these regions enclose specific parts of the car in (b), such as the front wheel, wheel arch and parts of the front door. The front wheel is likely to have a distinctive histogram, as it has a circular tyre surrounding the hubcap. This gives a circular edge which has a consistently changing orientation and a large gradient magnitude (as the tyre is black and the hubcap is grey), producing a histogram with a similar magnitude in each bin. The same region in the background image contains a straight edge, which would produce a histogram with a spike in one bin. This difference in histograms helps the GP tree to distinguish these two classes.

```
(- (* (+ (- (- (bin (HoG image 62,25 rectangle[16,4]
11) 5) 0.3114006771781549) (distance (HoG image 8,4
rectangle[55,15] 17) (HoG image 71,26 rectangle[91,30]
24))) (bin (HoG image 39,39 circle 15) 2)) (distance
(HoG image 50,24 column 12) (HoG image 72,31 circle
4))) (bin (HoG image 56,29 circle 17) 0))
```

(a)



(b) (c)

Fig. 5. Example program 3: 96 % training and 97 % test performance on UIUC dataset (a) Lisp expression, (b) Car (+ve) image, and (c) Background (−ve) image.

7 Conclusions

We proposed a new GP method for simultaneous region selection, feature construction and classification, which combines novel functions inspired by the HoG algorithm with the region selection concept proposed in the 2TGP method. Performance evaluation showed good results using the proposed method compared to the performance of 2TGP on three datasets with increasing difficulty. Performance was also promising when compared to other machine learning baselines using SURF features. The analysis of high-performing solutions showed that the GP-HoG approach could perform very well using simple programs. The adaptation of HoG for use as a GP function was shown to be an effective method of performing high-level feature extraction directly within a GP tree.

In the future, we would like to study the use of the extracted and constructed features produced by GP-HoG across different classifiers. This will help in identifying whether GP-HoG can be used for automatic feature extraction and construction, and whether these features are biased towards a specific type of classifiers. Another very important direction is to investigate the possibility of using other algorithms which could also be adapted directly in GP functions. For example, deeper analysis of the SURF or SIFT algorithms could produce functions that could be added to the GP-HoG approach. Other techniques such as edge detection could also be used within a GP tree in order to build a multi-faceted classifier which draws upon a range of techniques for complex classification tasks.

References

1. Agarwal, S., Awan, A., Roth, D.: Learning to detect objects in images via a sparse, part-based representation. IEEE Trans. Pattern Anal. Mach. Intell. **26**(11), 1475–1490 (2004)
2. Al-Sahaf, H., Song, A., Neshatian, K., Zhang, M.: Two-tier genetic programming: towards raw pixel-based image classification. Expert Syst. Appl. **39**(16), 12291–12301 (2012)
3. Al-Sahaf, H., Zhang, M., Johnston, M.: Genetic programming evolved filters from a small number of instances for multiclass texture classification. In: Proceedings of the 29th International Conference on Image and Vision Computing New Zealand, pp. 84–89. ACM (2014)
4. Al-Sahaf, H., Zhang, M., Johnston, M.: Binary image classification: a genetic programming approach to the problem of limited training instances. Evolutionary Computation (Journal, MIT Press) (2015). doi:10.1162/EVCO_a_00146
5. Al-Sahaf, H., Zhang, M., Johnston, M., Verma, B.: Image descriptor: a genetic programming approach to multiclass texture classification. In: Proceedings of 2015 IEEE Congress on Evolutionary Computation, pp. 2460–2467. IEEE (2015)
6. Back, T., Fogel, D.B., Michalewicz, Z.: Handbook of Evolutionary Computation. IOP Publishing Ltd., Bristol (1997)
7. Cheng, F., Yu, J., Xiong, H.: Facial expression recognition in JAFFE dataset based on gaussian process classification. IEEE Trans. Neural Netw. **21**(10), 1685–1690 (2010)
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 886–893. IEEE (2005)
9. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer Science & Business Media, Heidelberg (2003)
10. Espejo, P.G., Ventura, S., Herrera, F.: A survey on the application of genetic programming to classification. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. **40**(2), 121–144 (2010)
11. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explor. Newsl. **11**(1), 10–18 (2009)
12. Huang, Y., Wu, Z., Wang, L., Tan, T.: Feature coding in image classification: a comprehensive study. IEEE Trans. Pattern Anal. Mach. Intell. **36**(3), 493–506 (2014)
13. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
14. Krawiec, K.: Genetic programming-based construction of features for machine learning and knowledge discovery tasks. Genet. Program. Evolvable Mach. **3**(4), 329–343 (2002)
15. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the International Conference on Computer Vision, pp. 1150–1157. IEEE (1999)
16. Lyons, M., Akamatsu, S., Kamachi, M., Gyoba, J.: Coding facial expressions with gabor wavelets. In: Proceedings of the 3rd International Conference on Face & Gesture Recognition, pp. 200–205. IEEE (1998)
17. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Mach. Intell. **27**(10), 1615–1630 (2005)

18. Montana, D.J.: Strongly typed genetic programming. Evol. Comput. **3**(2), 199–230 (1995)
19. Nene, S.A., Nayar, S.K., Murase, H.: Columbia Object Image Library (COIL-20). Technical report (1996)
20. Perez, C.B., Olague, G.: Evolutionary learning of local descriptor operators for object recognition. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pp. 1051–1058. ACM (2009)
21. Poli, R.: Genetic programming for feature detection and image segmentation. In: Fogarty, T.C. (ed.) AISB-WS 1996. LNCS, vol. 1143, pp. 110–125. Springer, Heidelberg (1996)
22. Saini, R., Dutta, M.: Image segmentation for uneven lighting images using adaptive thresholding and dynamic window based on incremental window growing approach. Int. J. Comput. Appl. **56**(13), 31–36 (2012)
23. Shao, L., Liu, L., Li, X.: Feature learning for image classification via multiobjective genetic programming. IEEE Trans. Neural Netw. Learn. Syst. **25**(7), 1359–1371 (2014)
24. Winkeler, J.F., Manjunath, B.: Genetic programming for object detection. In: Proceedings of the Second Annual Conference on Genetic Programming, pp. 330–335. Morgan Kaufmann (1997)
25. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
26. Zhang, M., Ciesielski, V., Andreae, P.: A domain-independent window approach to multiclass object detection using genetic programming. EURASIP J. Adv. Signal Process. **2003**(8), 841–859 (2003)
27. Zhao, Y., Zhang, Y., Cheng, R., Wei, D., Li, G.: An enhanced histogram of oriented gradients for pedestrian detection. IEEE Intell. Transp. Syst. Mag. **7**(3), 29–38 (2015)