A Hybrid Genetic Programming Approach to Feature Detection and Image Classification

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
Email: {Andrew.Lensen, Harith.Al-Sahaf, Mengjie.Zhang, Bing.Xue}@ecs.vuw.ac.nz

***Abstract*—Image classification is a crucial task in Computer Vision. Feature detection represents a key component of the image classification process, which aims at detecting a set of important features that have the potential to facilitate the classification task. In this paper, we propose a Genetic Programming (GP) approach to image feature detection. The proposed method uses the Speeded Up Robust Features (SURF) method to extract features from regions automatically selected by GP, and adopts a wrapper approach combined with a voting scheme to perform image classification. The proposed approach is evaluated using three datasets of increasing difficulty, and is compared to five popularly used machine learning methods: Support Vector Machines, Random Forest, Naive Bayes, Decision Trees, and Adaptive Boosting. The experimental results show the proposed approach has achieved comparable or better performance than the five existing methods on all three datasets, and reveal its capability to automatically detect good regions from a large image from which good features are automatically constructed.**

I. Introduction

Feature detection in the image analysis domain seeks to extract patterns from an image that can then be used as a partial representation of an image's structure [1]. Techniques are generally categorised as *global* or *local* feature detection [1]. Global features are extracted from an image in its entirety, and as such are most useful when an image is being considered as a whole or has very distinct properties. Common techniques include colour-based approaches using histograms [2], texture-based features such as those produced by the Gabor filter [3] and eigenspace matching [4].

A local feature is extracted from an area of an image that differ from its immediate neighbourhood [1]. This difference can be as simple as a different colour or intensity, or more complex such as being a corner or an edge of an object. Local features, also known as keypoints, have several benefits over global features; they can be used to identify objects within images that occur at different locations (or even different scales or rotations), and also to identify multiple different objects in an image. As local feature extraction considers only a small part of the image, the overall composition of an image is also not critical.

One of the most widely used local feature extractors is the Scale-Invariant Feature Transform (SIFT) algorithm [5]. SIFT selects keypoints, i.e., local features, that are areas of an image with high contrast to their surroundings. The design of SIFT means that the keypoints are both scale and rotation-invariant. This makes SIFT a robust algorithm as it does not have the complications associated with other methods where the scale or angle of objects in images cannot be guaranteed. Speeded Up Robust Features (SURF) [6] is another scale and rotation invariant feature extractor that is generally faster than SIFT while achieving similar or better performance [6]. SURF uses a Hessian matrix for blob detection that is faster than the Difference of Gaussians (DoG) approach used in SIFT.

Many different approaches have been proposed using SIFT or SURF features for classification in images. One common approach is to use a *bag-of-keypoints* [7]. Keypoints are extracted from an image using SIFT or SURF, and are then clustered using k-means clustering. A histogram of cluster memberships is then produced with length k. The value at bin i is the number of keypoints in the image that are part of cluster i. The histogram is then classified using a classifier such as Support Vector Machines (SVM), Naive Bayes or Adaptive Boosting (AdaBoost) [7], [8]. However, this approach has a limitation that learning algorithms traditionally require fixed length features as input, for example an SVM cannot train effectively when a varying number of features is used. Farquhar *et al.* [9] proposed improvements to overcome this limitation by training to be performed directly on input features without any information being lost.

Evolutionary Computation (EC) techniques have also been applied to image feature extraction and classification problems. EC techniques have the ability to give good results by widely exploring a search space for a problem and considering solutions that the hand-crafted feature extraction methods may not. Genetic Programming (GP) [10] is an EC technique that mimics Darwin principles for nature selection and survival of the fittest. The GP search process starts by randomly generating a population of solutions (computer programs), and gradually evolves those solutions over a predefined number of iterations (generations) using genetic operators (crossover, mutation and reproduction). GP has been used in this domain since its introduction [11], [12].

One common technique uses images as the input to the program tree with functions which operate on individual pixels

978-1-5090-0357-0/15/$31.00 ©2015 IEEE

in the image. Pixel-based methods struggle for recognition of complex (e.g. many real world) images due to functions operating on a single or small number of pixels. A refinement to this idea is to use some simple pre-defined features which have the potential to capture useful information in an image. One approach [13] used the mean and standard deviation across some pre-defined regions as terminals in a GP tree. This produced trees that had potential to be understood by humans, while achieving good performance on two of the three datasets. The approach, however, struggled to produce programs that could be interpreted on the difficult retina dataset. Using pre-defined regions for feature extraction is somewhat inflexible and so can give poor performance for some problems when important parts of an image do not fall entirely within a pre-defined region.

A two-tier approach [14] was proposed which used GP to both select good regions of an image and also to extract features from the regions (in the form of pixel statistics) and perform classification. This approach achieved good performance across a range of datasets in different domains, and also produced easily interpretable solutions; one GP tree contained a circular region enclosing the eye of a face dataset, suggesting that the eye pixels are an important part of an image for classifying it as a face. This two-tier approach used simple pixel statistics to extract feature values from regions, which is a relatively low-level feature extraction technique. Using higher-level features could improve performance further, especially in difficult problems or where illumination, scale or rotation of objects vary. In this paper, we propose a method which uses a similar region selection strategy, but uses more advanced SURF features to improve classification performance.

Other EC techniques such as Particle Swarm Optimisation [15] and Genetic Algorithms (GAs) [16] have also been applied to image classification with some success. Bala *et al.* [16] used GAs to select features from images that would be classified by a decision tree. By doing so, they were able to exclude poor features from being used by the decision tree classifier. Combinations of EC techniques with other methods have also been shown to be effective at image classification [17], [18].

Both SIFT and SURF extract features from the entire image, which can be time consuming as computation time increases proportionately with the image size. Moreover, the extracted feature vectors can contain redundant or irrelevant features, which negatively affect the classification performance [19]. Applying SIFT or SURF to only sub-regions of the image has the potential to give more time-efficient and accurate classification by reducing the number of these problematic features.

A. Goals

This study aims to improve domain-independent object classification in images by using GP techniques. The proposed method uses GP to select good regions of the instance being evaluated, extracts good keypoints from those regions using SURF, classifies each keypoint using a wrapped classifier (e.g. SVM), and predicts the class label with a voting scheme. Specifically, we are interested in achieving the following objectives:

- Designing a program representation that is capable of detecting sub-regions of the image that are rich in features;
- Constructing a classification system to extract features from the selected regions and then use a SVM classifier and voting scheme to predict the class label;
- Evaluating the proposed method and comparing its performance against five well-known classifiers on three datasets of increasing difficulty and for different applications; and
- Investigating whether the regions detected by the new method are similar to those designed by domain experts.

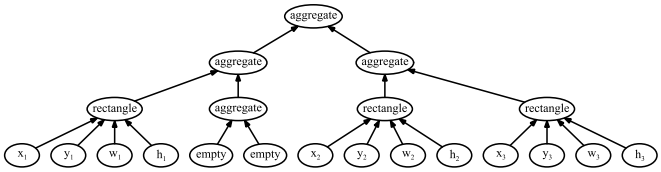


Fig. 1. An example of the general GP program structure.

B. Organisation

The rest of the paper is organised as follows. Section II discusses the proposed method and highlights its main components. The experiment design and evaluation process are explained in Section III. Experiment results are discussed in Section IV. Section V concludes the paper and suggests directions of future work.

II. The Proposed Method

This section discusses the proposed method and explains its main components. The discussion includes the GP program representation (i.e. the terminal and function sets), fitness function, process to generate SURF keypoints, and the class label predicting procedure.

A. GP Program Representation

As shown in Fig. 1, a tree-based GP structure is used to represent an evolved program in the proposed method. Due to having different numbers and types of input and output parameters, *Strongly-typed GP* (STGP) [20] is adopted to provide constraints for different nodes. In GP, the system uses two sets to generate an individual: (1) a terminal set; and (2) a function set. The terminal set is used to populate the terminal or leaf nodes of the program tree, whereas the internal nodes are populated using the function set.

The terminal set comprises of five elements. The x and w nodes are randomly generated integer values in $[0, min_{\text{width}}]$ where min_{width} is the minimum width of any image in the training set. Similarly, the y and h nodes are randomly generated integer values in $[0, min_{\text{height}}]$ where min_{height} is the minimum height of any image in the training set. The $empty$ node returns an empty list in order to allow the GP tree to terminate correctly.

The function set consists of two distinct nodes. The first is the $rectangle$ node that takes four arguments: x, y, w, and h (from the terminal set). It returns a list containing a single rectangle region starting at the coordinates (x, y) with width and height given by the w and h values respectively. The second node of the function set is the $aggregate$ node that takes two lists of regions, and returns a unified list. This node allows the system to produce multiple regions.

B. The Fitness Function

The fitness function is used to evaluate the performance of an evolved GP program on a set of images. This is used during the training phase to guide the GP method to select the best programs to evolve, and during the testing phase to measure the performance of the best evolved program on the unseen data. Picking a good fitness function is essential in achieving good results with GP as it determines the criteria that the evolutionary process is aiming to optimise. In this study, the datasets are balanced; nearly an equal number of instances from both classes is used to form each of the training and test sets. Hence, the fitness function used is just simple accuracy that is formally defined as:

$$fitness = \frac{N_{\text{correct}}}{N_{\text{total}}} \tag{1}$$

where N_{correct} is the number of instance correctly classified, and N_{total} is the total number of instances being evaluated. A fitness value of 1 represents the perfect fitness, a fitness value of 0 is the worst possible performance.

C. Generating SURF Keypoints

The SURF descriptor will generate varying numbers of keypoints based on the number of interest points an image has. This presents a problem for many classifiers where a static number of features is expected. To address this problem, we have developed the method presented in Algorithm 1. This method works by altering the Hessian threshold that SURF uses to determine how large the output from the Hessian filter must be in order to consider a point to be an interest point. The algorithm adjusts this threshold using a binary search until a predefined number of keypoints are retrieved. Using a fixed number of keypoints allows more effective training as a solution can perform consistently across a dataset and the situations where there are missing or extra keypoints are prevented.

Algorithm 1 Selecting p best keypoints

```
1: function FINDKEYPOINTS(image, lowerBound, upperBound, p)
2:     threshold ← (upperBound − lowerBound)/2 + lowerBound
3:     keypoints ← SURF(threshold)
4:     if |keypoints| = p then
5:         return keypoints
6:     else if |keypoints| > p then
7:         lowerBound ← threshold
8:     else
9:         upperBound ← threshold
10:    end if
11:    return FINDKEYPOINTS(image, lowerBound, upperBound, p)
12: end function
```

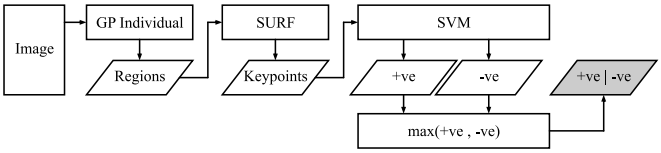


Fig. 2. The process of predicting the class label for an instance.

D. Predicting the Class Label

The class label for the instance being evaluated by an evolved program is predicted as follows. The program is evaluated, giving a list of regions as the output of the tree. Each region is then fed into SURF and p keypoints are generated, each being a vector of 64 values. The list of all keypoints extracted from all regions are then individually fed into an SVM classifier. The SVM classifier assigns a class label (either $+$ve or $-$ve) for each keypoint independently of all other points. The class label of a keypoint represents a single vote. The dominating class, i.e., the class with the most votes, is then returned as the predicted label for the instance being evaluated. This process is depicted in Fig. 2.

During training, an individual is trained by feeding all keypoints of all images into an SVM, where each keypoint is labelled with the class of the image it was extracted from. In other words, a list of pairs $(\vec{u}_i^j, y^j)$ of length $t \times r^j$ are fed into the SVM, where t is the number of instances in the training set, r^j is the total number of extracted keypoints from the j^{th} instance, $\vec{u}_i^j$ is the i^{th} keypoint's vector of the j^{th} instance, and y^j is the class label of the j^{th} instance. Each vector $\vec{u}_i^j$ consists of 64 values, i.e., $\left[a_{i,1}^j, \ldots, a_{i,64}^j\right]$ where $a_{i,s}^j$ is the s^{th} element in the i^{th} keypoint of the j^{th} instance. This ensures that the SVM trains on each keypoint independently. This is important for generalising performance as a voting approach is being used.

It is highly likely that GP generates regions which exceed the image boundaries. These regions are ignored, and so play no part in the classification process. A number of $empty$ nodes can be part of the evolved program; they are also ignored. Similarly, regions which reach 100 iterations of Algorithm 1 without producing p keypoints are discarded, as it is unlikely the correct number of keypoints will be produced with additional iterations. Limiting the number of iterations avoids an overly long training time.

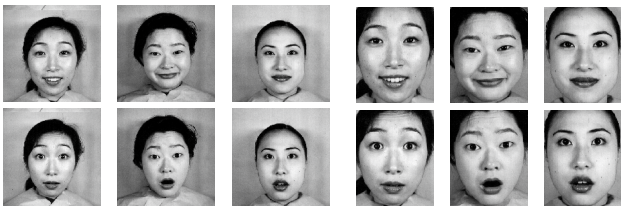
III. Experiment Design

The datasets used to evaluate the proposed method are discussed in this section. This section also lists and explains the baseline methods, as well as discussing the evolutionary parameter settings and libraries used in our implementation.

A. Evaluation Data Sets

Three datasets are used in this study to evaluate the proposed method. The datasets vary in difficulty and type (i.e. application). Each of the datasets is comprised of two classes (binary classification) and contains grey-scale images.

Fig. 3. Samples of the UIUC cars dataset shows instances of the car and background classes in the first and second row respectively.



(a) (b)

Fig. 4. Samples of the JAFFE dataset (a) before cropping, and (b) after cropping.

The first dataset is the *UIUC database for Car Detection*[1] [21] dataset. The dataset consists of $1,050$ instances that 550 are cars and 500 are background. Each of the car instances shows the side view of the vehicle captured from the same angle and distance (giving the same scale) as shown in Fig. 3. The instances in this dataset are 100×40 pixels in size.

The *Japanese Female Facial Expression*[2] (JAFFE) [22] dataset is used in the literature to identify different facial expressions. There are 213 images in this dataset that fall into 7 classes: neutral, happy, sad, surprised, angry, disgust, and fear. Ten Japanese female subjects provided several images for each facial expression. Our second dataset uses the instances of the happy and surprised classes as shown in Fig. 4. Similar to [23], we have cropped the instances of this dataset to remove the image background and most of the subject's hair, leaving only the face. Each instance is grey-scale and after cropping has dimensions ranging between 121 and 143 in width, and between 164 and 207 in height.

The two classes of the third dataset are drawn from a popular and widely used dataset in computer vision called the *Columbia Object Image Library*[3] (COIL-20) [24]. The COIL-20 dataset consists of 20 classes each of which represent a different toy object such as cars, rubber ducks, and cups. Each object was placed on a turntable in a scene with a black background. The object was rotated through $360°$, with an image captured every $5°$ of rotation, giving 72 images per object. The images were normalized to be 128×128 pixels and the object was centred in the images as much as possible. The colour of each image was also normalised by making the brightest pixel equal to 255 and scaling the other pixel values accordingly. The *car* and *rubber duck* classes are used to form our third dataset in this study. Fig. 5 shows some examples of these two classes.

The performance on these datasets was evaluated by using

[1]Available at: http://cogcomp.cs.illinois.edu/Data/Car/

[2]Available at: http://www.kasrl.org/jaffe.html

[3]Available at: http://www.cs.columbia.edu/CAVE/software/

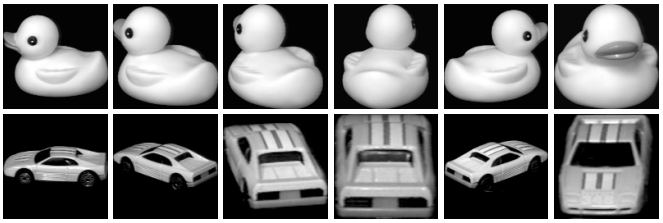


Fig. 5. Samples of the COIL-20 dataset, where instances of the cars and rubber ducks classes are shown in the first and second row respectively.

k-fold cross validation. The instances of the UIUC and COIL-20 datasets were randomly split into 10 folds. The JAFFE dataset is smaller and has 3 images of the same person for each expression, therefore it was manually split into 3 folds so that images of each person occurred across each fold.

B. Baseline Methods

The performance of the proposed method is assessed by comparing it to five well-known machine learning algorithms which are detailed in [25]:

- Support Vector Machine (SVM): a powerful classification model that analyses data and recognises patterns using a learning algorithm. The *Sequential Minimal Optimisation* (SMO) [26] learning algorithm was used in this study.
- Naive Bayes (NB): a simple probabilistic method which follows Bayes' theorem to build a model.
- Adaptive Boosting (AdaBoost): an adaptive method that builds a model by improving on misclassified instances of previous models.
- Decision Trees (J48): a re-implementation of the C4.5 decision tree learning algorithm.
- Random Forest (RF): an ensemble method that constructs a set of decision trees, which is designed to mitigate the overfitting habit of decision trees.

SURF produces a feature vector consisting of 64 values (i.e. features) for each keypoint, hence, there will be $p \times 64$ features per instance where p is the number of keypoints used. In each of the baseline methods, the training set contains instances represented as pairs of concatenated SURF keypoints and the class label for the instance. The training set is formally represented as follows: $\left[\left(\vec{u}_1^1 \ldots \vec{u}_p^1, y^1\right), \left(\vec{u}_1^2 \ldots \vec{u}_p^2, y^2\right), \ldots, \left(\vec{u}_1^t \ldots \vec{u}_p^t, y^t\right)\right]$ where y^i is the class label of the i^{th} instance, t is the total number of instances in the training set, and each vector $\vec{u}_i^j$ consists of 64 values, as defined in Section II-D.

C. Evolutionary Parameters and Implementation

The GP system is evolved until 50 generations are completed or an ideal individual (with 100% accuracy) is found. The population is relatively large with $1,024$ individuals, and the initial population is generated using the ramped-half-and-half method. The diversity of the population has been maintained by using a 30% mutation rate. In order to maintain the progress of the evolutionary process, the "keep top n individuals" scheme is adopted with $n = 10$. The crossover

TABLE I
THE GP EVOLUTIONARY PARAMETERS

Parameter	Value	Parameter	Value
Generations	50	Crossover Rate	0.70
Population Size	1024	Mutation Rate	0.30
Minimum Depth	2	Elitism	keep the top 10
Maximum Depth	6	Selection Type	Tournament
Initial Population	Half-and-half	Tournament size	7

TABLE II
THE ACCURACIES ON THE UIUC DATASET.

	$p = 5$		$p = 10$		$p = 20$		$p = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
GP	0.99	0.92	0.98	0.92	N/A	N/A	N/A	N/A
SVM	0.99	0.92	1.00	0.91	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
J48	0.99	0.84	0.99	0.83	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
AdaBoost	0.88	0.85	0.88	0.84	N/A	N/A	N/A	N/A

operator plays an essential role in the system and so is used more often (70% of the time) than the mutation operator. Table I summarises these parameters.

As GP is a stochastic method, each experiment is run 35 times with different fixed seeds. For each fold in a run, the best evolved program on the training set is evaluated on the test set. The performance of the run is then the average across all k-folds of that run. The results reported in Section IV are the average performance across the 35 runs.

The proposed method is implemented using the STGP platform provided by the *Java-based Evolutionary Computation Research System* (ECJ) package [27]. The *Waikato Environment for Knowledge Analysis* (WEKA) [25] implementations of SVM, J48, NB, RF, and AdaBoost are used for the baseline approaches.

IV. RESULTS AND DISCUSSION

The results are presented and discussed in this section. A program evolved by the new method is also briefly analysed.

The results of each dataset are presented in a table that horizontally consists of five blocks. The method names are shown in the first block and each subsequent block lists the average performance of the methods on the training and test sets across all folds, using different numbers of SURF keypoints (i.e. 5, 10, 20, and 50).

Table II presents the results on the UIUC dataset. The values of the last two blocks ($p = 20$ and $p = 50$) are not available, as SURF could not generate this many keypoints due to lack of interest points. The proposed method shows promising results for the $p = 5$ and $p = 10$ blocks, with performance that is very close to RF which is the best baseline method. This suggests that the proposed method can compete with an ensemble or committee based approach where more than one classifier contributes towards predicting the class label.

The results on the JAFFE dataset are shown in Table III. The results are consistently good across a range of quantities of keypoints. The proposed method has outperformed all baseline

TABLE III
THE ACCURACIES ON THE JAFFE DATASET.

	$p = 5$		$p = 10$		$p = 20$		$p = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
GP	1.00	**0.87**	1.00	**0.87**	1.00	**0.89**	1.00	**0.89**
SVM	1.00	0.63	1.00	0.72	1.00	0.74	1.00	0.82
NB	0.89	0.72	0.92	0.72	0.98	0.75	1.00	0.69
J48	0.98	0.70	0.98	0.56	0.98	0.59	0.98	0.79
RF	1.00	0.77	1.00	0.76	1.00	0.77	1.00	0.71
AdaBoost	1.00	0.70	1.00	0.77	1.00	0.67	1.00	0.71

TABLE IV
THE ACCURACIES ON THE COIL-20 DATASET.

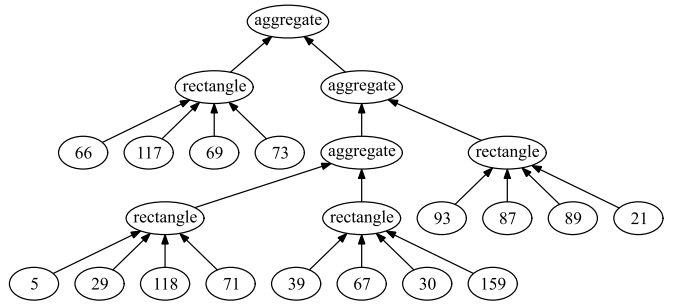
	$p = 5$		$p = 10$		$p = 20$		$p = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
GP	1.00	0.99	1.00	0.99	1.00	1.00	1.00	0.99
SVM	1.00	0.99	1.00	0.99	1.00	0.99	1.00	1.00
NB	0.99	0.97	0.99	0.96	1.00	0.91	1.00	0.94
J48	0.99	0.86	0.99	0.86	0.99	0.86	0.99	0.85
RF	1.00	0.99	1.00	0.99	1.00	1.00	1.00	1.00
AdaBoost	1.00	0.96	1.00	0.96	1.00	0.93	1.00	0.93

methods on this dataset with a minimum gap of 7% on the test set when compared to SVM when 50 keypoints is used. The improved performance compared to the baselines indicates the ability of this method to detect very important regions (as discussed in the following subsection).

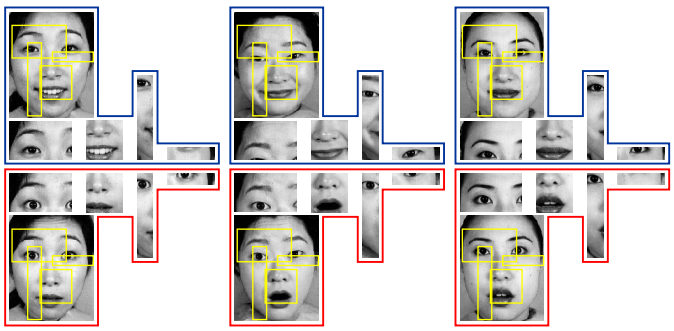
As shown in Table IV, the proposed method performs equivalently to the best baseline methods (SVM, RF) on the COIL-20 dataset. As this dataset is fairly easy, the proposed and baseline methods were both able to achieve near perfect performance. This shows that the additional steps in the proposed method (as compared to the baseline SVM) do not negatively affect performance on easy datasets, suggesting that it is good across a range of datasets and applications.

A. Further Analysis

Fig. 6(a) shows the tree representation of a program evolved on the JAFFE dataset. This program achieved on average over 95% accuracy on the unseen data across the 3-fold cross validation. The program detects four interesting regions that are highlighted in yellow on six instances that have been drawn from both "happy" and "surprised" classes (three instances of each class) as presented in Fig. 6(b)-(d). To ease the visual comparison between instances of the same subject in the two classes, those regions are also cropped. Each instance and cropped regions of the happy class are bounded in blue, whereas for the surprised class, they are bounded in red. The cropped regions clearly show that the program has detected regions of the eyes and mouth. The eyes appear more flat when the subject is happy due to human nature in pushing-up the cheeks while smiling, whereas the eyes are opened wide when the subjects are surprised. The mouth is upturned or shows a bright region due to the teeth being visible while smiling, whilst it is darker (teeth are covered by lips) and flatter when the subject is surprised. By focusing on these regions where there are distinct differences, it is likely that more useful features are extracted for better classification performance.



(a)



(b) (c) (d)

Fig. 6. Example shows (a) the tree representation of a program evolved on the JAFFE dataset; and (b)-(d) the detect regions on three samples.

V. Conclusions

In this paper, a GP method for feature detection in images is proposed. The proposed method evolves a program that detects sub-regions of an image, extracts SURF points from those regions, and classifies the instances being evaluated using a SVM classifier and a voting scheme. Using three datasets of varying difficulty and from different domains, the new method achieves comparable or better performance compared to well-known machine learning classifiers. Furthermore, the proposed method has detected regions similar to those designed by domain experts which indicates the capability of this method to identify such important regions. However, there is still plenty of space for improving the effectiveness of the proposed method. Applying this method to multi-class classification tasks is one area of improvement we would like to investigate. We also would like to extend this method by changing the program representation to automatically evolve a classifier using GP instead of using a wrapped classifier in the hope that it will allow even more effective training of the system.

References

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

[2] A. K. Jain and A. Vailaya, "Image retrieval using color and shape," *Pattern recognition*, vol. 29, no. 8, pp. 1233–1244, 1996.

[3] B. S. Manjunath and W.-Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837–842, 1996.

[4] H. Murase and S. K. Nayar, "Visual learning and recognition of 3-D objects from appearance," *International journal of computer vision*, vol. 14, no. 1, pp. 5–24, 1995.

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

[6] 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.

[7] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on Statistical Learning in Computer Vision, ECCV*, vol. 1, no. 1. Springer, 2004, pp. 1–16.

[8] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2. IEEE, 2005, pp. 994–1000.

[9] J. Farquhar, S. Szedmak, H. Meng, and J. Shawe-Taylor, "Improving "bag-of-keypoints" image categorisation: Generative models and pdf-kernels," University of Southampton, Tech. Rep., 2005.

[10] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*. Lulu, 2008, (With contributions by J. R. Koza).

[11] R. Poli, "Genetic programming for feature detection and image segmentation," in *Evolutionary Computing*, ser. Lecture Notes in Computer Science. Springer, 1996, vol. 1143, pp. 110–125.

[12] M. Maghoumi and B. J. Ross, "A comparison of genetic programming feature extraction languages for image classification," in *Proceedings of the IEEE Symposium on Computational Intelligence for Multimedia, Signal and Vision Processing*. IEEE, 2014, pp. 1–8.

[13] 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.

[14] 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.

[15] M. Omran, A. Salman, and A. P. Engelbrecht, "Image classification using particle swarm optimization," in *Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning*, vol. 1. Springer, 2002, pp. 18–22.

[16] J. Bala, K. De Jong, J. Huang, H. Vafaie, and H. Wechsler, "Using learning to facilitate the evolution of features for recognizing visual concepts," *Evolutionary Computation*, vol. 4, no. 3, pp. 297–311, 1996.

[17] P. Ghamisi and J. Benediktsson, "Feature selection based on hybridization of genetic algorithm and particle swarm optimization," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 2, pp. 309–313, 2015.

[18] S. Li, H. Wu, D. Wan, and J. Zhu, "An effective feature selection method for hyperspectral image classification based on genetic algorithm and support vector machine," *Knowledge-Based Systems*, vol. 24, no. 1, pp. 40–48, 2011.

[19] R. Gonzalez Valenzuela, W. Robson Schwartz, and H. Pedrini, "Dimensionality reduction through PCA over SIFT and SURF descriptors," in *Proceedings of IEEE 11th International Conference on Cybernetic Intelligent Systems*. IEEE, 2012, pp. 58–63.

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

[21] S. Agarwal, A. Awan, and D. Roth, "Learning to detect objects in images via a sparse, part-based representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1475–1490, 2004.

[22] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets," in *Proceedings of the 3rd International Conference on Face & Gesture Recognition*. IEEE, 1998, pp. 200–205.

[23] F. Cheng, J. Yu, and H. Xiong, "Facial expression recognition in JAFFE dataset based on gaussian process classification," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1685–1690, 2010.

[24] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia Object Image Library (COIL-20)," Tech. Rep., 1996.

[25] 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.

[26] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. MIT Press, 1999, pp. 185–208.

[27] S. Luke, *Essentials of Metaheuristics*, 2nd ed. Lulu, 2013. [Online]. Available: http://cs.gmu.edu/$\sim$sean/book/metaheuristics/