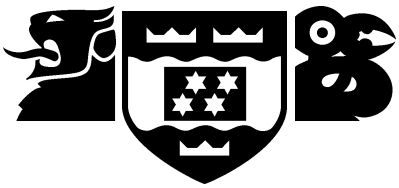
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui



School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Genetic Programming for Image Classification using High-Level Features

Andrew Lensen

Supervisors: Mengjie Zhang, Harith Al-Sahaf, Bing Xue

Submitted in partial fulfilment of the requirements for Bachelor of Science with Honours in Computer Science.

Abstract

Image analysis is a key area in the computer vision domain that has many applications. Genetic Programming (GP) has been applied to this area extensively, with positive results. High-level features extracted from methods such as Speeded Up Robust Features (SURF) and Histogram of Orientated Gradients (HoG) are commonly used for object detection using machine learning techniques. However, GP techniques are not often used with these methods, despite being applied extensively to image analysis problems. This work investigates several novel approaches for using GP with high-level features for image classification. These new approaches are 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. The use of GP for feature extraction and construction is also investigated, achieving high performance using only a few constructed features.

Acknowledgments

First and foremost, I must thank my supervisors: Prof. Mengjie Zhang, for his motivation and excellent guidance, Harith Al-Sahaf, for the many interesting and useful discussions and his ability to put things into perspective, and Dr. Bing Xue, for her ongoing support and detailed feedback. I feel very lucky to have had the help of three such caring and friendly individuals – this work is much more refined and thorough because of them.

My appreciation also goes to my family, who supported me throughout this year and provided an often-needed escape from study. In particular, I am grateful to my father who provided a caring and supportive environment to come home to.

Finally, my thanks to all my friends and fellow students who made this journey less lonely and who shared a common understanding of the difficulty that is an honours degree. The mostly study-related discussions and (infrequent) distractions made this experience all the more enjoyable.

Contents

Figures

Tables

Chapter 1

Introduction

The automated analysis of images by computers is an area of computer science with many interesting uses. Tasks such as Optical Character Recognition (OCR), facial identification, medical imaging analysis, and motion detection are all very useful in our modern world. While humans can analyse images with relative ease, it is far too labour-intensive to analyse a large number of images manually. For example, it would take many security guards to monitor all the cameras in an airport in order to attempt to detect certain individuals. With good image analysis software, a computer system could do this much more efficiently and precisely. However, image analysis remains a difficult problem in computer science as different problems often require different approaches in order to extract useful information from images. This work applies Artificial Intelligence techniques in a novel way with existing techniques to improve upon the accuracy of image classification.

1.1 Motivations

Genetic Programming (GP) has been applied extensively to image analysis problems [42, 50, 20] since it was introduced in the 1990s. Techniques generally use GP to extract *features* (numerical values which can be used for characterising an image) from raw images by using pixel statistics [48, 50], sliding window [45] or filter [3] approaches. While these approaches have had some success, there are a number of other feature extraction algorithms which produce better features in a more consistent manner.

Speeded-Up Robust Features (SURF) [8] and *Histogram of Oriented Gradients* (HoG) [12] are two popular high-level feature extraction algorithms. SURF produces *keypoints* which represent important points in an image, such as corners of objects. A keypoint is described using a keypoint descriptor which can be used as a feature vector to characterise an image. HoG produces a histogram of the magnitude of gradients within an image. A histogram can be used as a feature vector, where each bin in a histogram is a feature that can help classify an image. The similarity of two images can be found by comparing the similarity of their features, and images can be classified by using features as inputs to classifiers such as Support Vector Machines (SVMs) [38] and AdaBoost [38].

While GP, SURF, and HoG have each been used extensively for image classification, [34, 8, 26, 13] literature combining the SURF and HoG high-level feature extractors with GP is limited. GP has been extensively applied to extract low-level features for image classification [50], but is not commonly used to classify high-level features extracted by other algorithms, despite having significant success in binary classification tasks [4]. Furthermore, the SURF and HoG algorithms can be limited by their approach that extracts features across a whole image, when only parts of an image may produce useful features. Combining GP with these feature extractors has the potential to improve image classification results.

1.2 Goals

This work aims to develop a new domain-independent approach to image classification by using GP in combination with the existing SURF and HoG high-level feature extraction algorithms. It aims to investigate using GP for classification of high-level features, as well as for extraction and construction of better features based on these algorithms. This will be done through a number of objectives:

1. Using a hybrid approach that combines GP and SURF to give a coherent classification system. Two potential methods for achieving this objected will be developed and investigated:
    - Using SURF to extract keypoints from raw images and then using GP to classify the image using features extracted from these keypoints as input. This method uses GP for classification.
    - Using GP to select regions from raw images and then using SURF to extract keypoints from these regions. The image will then be classified by using an existing classifier (e.g. SVM) using features extracted from the keypoints. Here, GP is used to produce better features by finding important regions of an image.
2. Using GP for simultaneous region selection, feature extraction and image classification. This approach will use new GP functions that are inspired by the HoG algorithm. This objective is a single-stage approach, as the ideas behind the HoG algorithm will be directly used within the GP tree as functions that can be chosen as part of the evolutionary process. As such, machine learning is applied to the entire classification system, giving the potential for improved classification accuracy as the whole system may be optimised in one process.
3. Using the features extracted from the new methods as inputs to other commonly used classification algorithms. This will test the performance of the new methods as feature extraction and construction algorithms. Other classification algorithms may perform better than GP on the extracted features, increasing classification performance.

The performance of these new approaches will be compared to a range of machine learning baseline methods across a range of datasets.

1.3 Major Contributions

This work contains a number of major contributions:

- This work shows how GP can be used for classification of high-level features (from SURF keypoints) using a voting system, where each keypoint is classified separately. Additionally, it is shown that this voting approach can be applied to other classifiers with good results.
- This work shows that GP can be used to automatically select small feature-rich regions of interest from large images. These regions can then be used with the SURF algorithm to produce features for image classification with an SVM, improving performance. A paper titled "A Hybrid Genetic Programming Approach for Feature Detection and Image Classification" that describes this contribution has been submitted to the 30th International Conference on Image and Vision Computing New Zealand (IVCNZ 2015).

- This work shows how GP can be used for simultaneously selecting important regions, extracting good features and performing classification by using a single evolved program that combines HoG-inspired features and low-level features. A paper titled "Image Classification using Genetic Programming for High-Level Feature Construction" is being prepared for EvoIASP 2016.
- This work shows that the programs produced by GP can be interpreted and understood by humans, which is a key goal in data mining.
- This work shows that the features extracted and constructed by GP can be used in other classification algorithms, allowing very accurate image classification with only a small number of features.

1.4 Report Organisation

Chapter 2 provides an overview of the image classification and Evolutionary Computation domains, and discusses the literature that is related to this work. Chapter 3 discusses the method, results and analysis of the new approach using GP and SURF (i.e. Hybrid GP-SURF), as well as the datasets and baseline methods used in this work. Chapter 4 presents the GP method using HoG-inspired features (i.e. GP-HoG) and analyses good programs generated by this method. Chapter 5 analyses how the features extracted and constructed by GP-HoG can be used with other classifiers. Conclusions and possibilities for future work are given in Chapter 6.

Chapter 2

Background

This chapter reviews some of the important areas and ideas that form the basis of this work. An overview of Machine Learning, Evolutionary Computation, Genetic Programming, performance evaluation principles and common image classification techniques is provided. Related work in the image classification field is also discussed, providing an understanding of the limitations of the literature that this work seeks to address.

2.1 Machine Learning

Machine learning [5] is a field of Artificial Intelligence (AI) which contains algorithms which are able to learn from data (i.e. *instances*), in order to make predictions on future data that they have not been exposed to. These algorithms often function by building and refining a model based on input data, rather than being manually crafted as is common with computer algorithms. Machine learning has been applied to a huge range of tasks, which can generally be categorised in one of three categories [44]:

- supervised learning (the expected outputs are known), e.g. labelling an image as "human" or "animal", when it has learnt on some images that already have these labels.
- unsupervised learning (the expected outputs are unknown), e.g. grouping people based on their interests in order to personalise film recommendations.
- reinforcement learning (the algorithm tries to maximise some "reward" without explicit feedback), e.g. a mouse trying to solve a maze, where the reward is how close it is to the end.

A large number of approaches have been proposed, each of which tends to be effective on a different group of tasks. Common categories of algorithms include decision trees, artificial neural networks, linear discriminant analysis models, Bayesian networks, clustering algorithms and evolutionary computation algorithms [5].

2.1.1 Supervised Learning

Supervised learning is perhaps the largest category of machine learning, as knowing the expected outputs for a set of instances is relatively common in the most popular tasks such as classification (labelling of instances) and regression (developing a mathematical function to map an input to an output). In supervised learning, a model will be trained on some known instances, called a training set, until a stopping criterion is met [44]. Common stopping criteria include a given accuracy being reached on the training set, or some amount

of computational time elapsing. The model is then applied to the instances in the test set, which have not been trained on at all, and so are unseen to the model. The accuracy of the model on the test set gives a representation of how well the model is expected to perform on future data, and hence gives a measure of how effective the model is.

One common issue in supervised learning is over-fitting, where a model learns too specifically on the training set, and so has poor performance on the test set where instances have slightly different characteristics [49]. A third dataset, called the validation set, can be used to address this problem. This dataset is not used to directly train the model either. After each iteration of the training process, the performance of the model can be evaluated on the validation set. When this performance begins to decrease, it suggests that the model is beginning to be over-trained, and so training should be stopped. Using a validation set allows test performance to be improved, by preventing the model from being too accurate (as to be overly specific) on the training set.

The use of a training and test set requires an adequate number of instances in order to ensure that an accurate and fair evaluation of the model can be obtained. If there is a small number of instances available, it is difficult to ensure that the small test set will give a fair evaluation of how well the model will perform on future data [49]. A small training set may also reduce the performance of a model, as it has fewer instances to train on, and has a higher chance of over-training. This issue is commonly addressed by using n-fold cross-validation [36]. This technique splits all the available instances evenly into n folds. Each fold then has a turn at being used for the test set, while all other folds comprise the training set. The performance of the model is then said to be the mean performance across all n folds. This ensures that all instances are able used as part of a test set, which increases the confidence that a model can generalise on unseen data.

2.2 Evolutionary Computation

Evolutionary Computation (EC) is a large field of AI which contains algorithms that are inspired by biological evolutionary principles [6, 14]. 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 each iteration in order to gradually improve solutions towards the optimal solution. Although they are known as global search methods, EC techniques do not guarantee finding an optimal solution, as they essentially use a guided random search process to increase the quality of solutions found within a reasonable amount of computational time.

EC techniques include Evolutionary Algorithms [14] (using Darwinian principles), Swarm Intelligence [25] (based on animal swarming behaviour and social intelligence) and many other algorithms. Evolutionary Algorithms (EAs) draw upon Darwinian principles such as reproduction, mutation, and crossover to evolve solutions to a problem in a way that mimics natural evolution [14]. An EA begins with some randomly generated solutions to a problem and then iteratively evolves these candidate solutions using these principles to develop progressively better solutions. In each iteration, the solutions are first evaluated to find their fitness, using a fitness function. The best solutions are chosen based on their fitness and have a genetic operator applied to them to attempt to improve them further. These operators include mutation (randomly change a solution), crossover (swap parts of two different solutions to try to combine good features of each) and reproduction (keep the very best solutions the same). By repeating this process it is hoped that solutions will continue to improve until one gives good results to the problem. This process terminates upon reaching a given criterion (e.g. a number of iterations or a good enough solution).

Two of the most common EC techniques are Particle Swarm Optimisation (PSO) [24] and Genetic Algorithms (GAs) [22]. PSO is a swarm intelligence technique that mimics the behaviour of swarming animals, such as birds, by using a number of particles which are moved around the search space. Each particle represents a candidate solution, where the solution is encoded within a particle. A particle's position is changed after each iteration based on its best known position, as well as the best known position in the swarm. GAs are an EA algorithm that use similar encoding methods to PSO, but use the biological principles discussed above to iteratively improve their pool of solutions.

2.3 Genetic Programming

Genetic Programming (GP) [28] is an EA based on GAs which models solutions in the form of computer programs. The most common representation is using a tree structure, where the root of the tree is the output of the genetic program and the leaves of the tree are inputs. 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. For example, a "+" node could take two integer inputs and output their sum. Terminal nodes have no children, and are the leaves of a tree. For example, the number 5 could be a terminal node which produces an integer. This representation is very similar to that of computer programs, which can often be understood more clearly when represented using a tree structure.

The same evolutionary process is applied in GP as in other EAs. An overview of this process is shown in Figure 2.1. The initial pool of candidate solutions can be created using a variety of methods [43]. The *full* method randomly selects functions to be nodes of a GP tree until a given depth is reached, and then randomly chooses terminals to form the leaf nodes. This approach ensures a full and balanced tree. The *grow* method randomly selects nodes from both the function and terminal sets as the tree is built. If a terminal node is chosen, the branch of the tree being built is terminated. This method gives a higher variety in tree structure, allowing a range of tree sizes. Another popular method is to use a combination of these two ideas, called the *ramped half-and-half* method, where half of the population uses the grow method and the other half uses the full method. This gives variety and diversity to the initial trees, while still insuring some large trees are included [43].

As tree-based GP uses a different structure than that of GAs, the genetic operators vary slightly in how they are designed [43]. The mutation operator selects a random subtree (by selecting a random node as the root of the subtree) and then replaces it with a new subtree

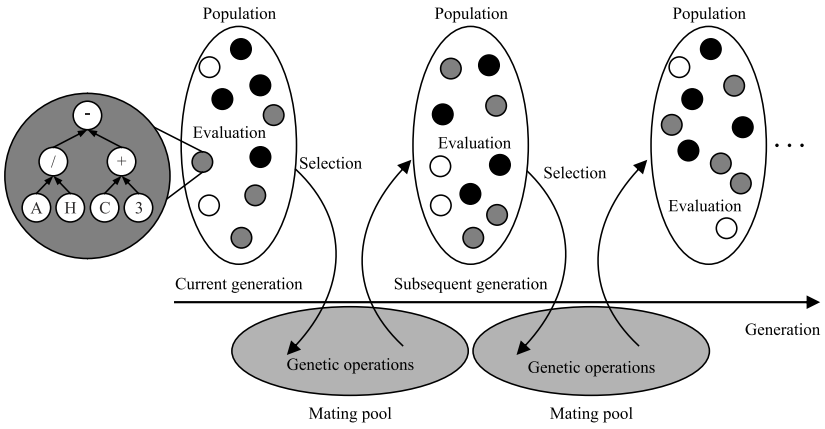


Figure 2.1: The GP evolutionary process

produced using one of the program generation methods discussed above. The crossover operator selects a random subtrees in two trees, and then swaps the two subtrees. The reproduction operator is unchanged – a given GP tree is preserved between generations.

The performance of an individual in GP during training is measured by the fitness function. A common fitness function for classification problems is the accuracy of the tree, i.e. how many instances it correctly classifies [34]. GP can be used for binary classification by applying a threshold to the output of the root of the tree; if the output is less than 0, then the instance is class 1, otherwise it is class 2 [4]. In this case, the terminals of the GP tree will often represent features of the instances being classified. For example, if GP was used to classify types of plants, one terminal may be the length of the plant's leaf. An instance can then be classified using GP by setting this terminal's value to the length of the instance's leaf, evaluating the tree and using the output of the root to give a classification label.

2.3.1 Strongly-Typed Genetic Programming

Strongly-Typed GP (STGP) [37] is a refinement of the standard GP technique which places restrictions on how functions and terminals may be combined. Standard GP requires only that each function has a certain number of children nodes (based on the design of that function); it does not place restrictions on what type of nodes those children must be. In STGP, the type of each child is specified in the function design. For example, in an image classification application, a function may take one child of type *image* and one child of type *integer*. This design ensures that functions will always have certain types of children, and hence will always output a certain type. This allows for more sophisticated programs to be generated, as more complex functions can be designed. This project uses STGP for this purpose, for example by designing some functions which operate on vectors and others which operate on single values. These more advanced programs can help to increase the performance of a GP system as more sophisticated GP trees are produced which can have better performance.

2.4 Performance Evaluation

A key consideration in machine learning is how the performance of the model is evaluated. There are three main attributes that determine the performance of a model:

- Effectiveness of the model: one common metric is to simply find the accuracy of the system on a previously-unseen test set. This gives an indication of how accurately the system will perform when it is used in the future [49]. While simple accuracy (percentage correct) is often sufficient, other measures such as the F-measure or Area Under the RoC Curve (AUC) [10] can be useful for measuring performance on unbalanced (unequal number of instances in each class) or difficult datasets.
- Efficiency of the model: the amount of computational time and resources required to apply a model to an instance can affect how useful it is in a real-world scenario [17]. While a long training time may often be acceptable, a trained system that takes several hours to evaluate an instance may be too slow to be useful, even if it is very accurate.
- Understandability of the model: a good system should also be easy to understand and interpret [17]; data mining tasks (such as image classification) should produce models that can be understood so they can be verified, trusted, and potentially manually improved. In terms of GP, an understandable solution would be an individual that has a tree with a relatively small depth and which has functions that have behaviour and inputs which can be easily reasoned about.

2.5 Image Classification

Image classification is an area of Computer Vision which is concerned with classifying an image with a label or class. While humans are able to consistently recognise images, computer algorithms have struggled to achieve similar performance due to a range of factors. A human is able to identify parts of an image that has been rotated, translated or changed in scale, but many algorithms will struggle to do so unless carefully designed. Differences in illumination or contrast across images also present challenges for accurate image classification. A popular approach to address these issues is to first extract some useful features from an image which provide a higher-level representation of the contents of an image than the raw image pixels [23]. Features can be designed to be rotation, translation, scale or illumination-invariant, which allows use across different images more accurately than when using image pixels. The extracted features are then fed into a classifier, which provides a classification of the whole image.

One common technique for feature extraction is the extraction of descriptors which represent local interest regions within an image [35]. By combining multiple descriptors, an accurate representation of an image can be obtained, which can then be used for image classification [35]. One popular technique that uses this concept is the *Scale-Invariant Feature Transform* (SIFT) [31] algorithm which addresses common problems such as scale and illumination. SIFT extracts *keypoints* which are pixels with higher contrast than their neighbours. A keypoint can then be described using a feature vector. While SIFT is an improvement over simpler methods, it still suffers from performance issues in some circumstances [35]. SURF [8] is another feature extraction algorithm which is partially inspired by SIFT. SURF is similar to SIFT, but is claimed to be several times faster and more robust against different image transformations [8]. SURF uses a Hessian matrix for blob detection that is faster than the Difference of Gaussians approach used in SIFT.

The Histogram of Oriented Gradients (HoG) [12] 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. Each pixel contributes to the block's histogram based on the magnitude of the gradient (how much weight to add to the histogram) and its orientation (which bin of the histogram it belongs to). 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. As with SURF and SIFT, the HoG histogram can then be used with a classifier to identify the class of an image.

2.6 Related Work

This section discusses some of the previous work on image classification, with a particular focus on techniques using GP, SURF or HoG. Recent work using GP for feature selection and construction is also discussed.

2.6.1 Genetic Programming for Image Classification

GP has been applied in a number of ways for image classification [34]. One common technique uses images as the input to the program tree with functions which operate on individual pixels 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 [50] used the mean and standard deviation across some pre-defined regions as terminals in a GP tree. This produced trees that had the potential to be understood by humans, while achieving good performance on two of the three datasets. This 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.

Bhowan *et al.* [9] proposed new techniques to improve the performance of GP when performing image classification on an unbalanced dataset. GP can struggle to perform well on unbalanced problems, as a simple fitness function such as accuracy will be influenced much more by the accuracy of the larger class. To address this, the authors proposed a range of new fitness functions which weighted each class more fairly. They also proposed a multi-objective approach, which addresses this issue by using the accuracy of each class as a separate objective to be optimised. These techniques showed a performance improvement over the baseline GP approach.

A two-tier approach [2] 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 varies.

Another method used GP to enhance the performance of the existing high-level SIFT feature extractor by producing more refined keypoints for classification [21]. While this technique showed improved performance for specific problems, it did not increase performance in the general case. This method did not attempt to use GP for classification of the SIFT keypoints, which is an approach that could be promising as a range of functions can be applied to a keypoint descriptor.

2.6.2 Other Image Classification Techniques

Other EC techniques such as PSO and GAs have also been applied to image classification with some success. Omran *et al.* introduced a PSO approach which performed unsupervised classification of individual pixels in an image [41]. Pixels were clustered together, with each cluster representing a single class. Particles were encoded as a series of mappings from pixels to clusters. PSO was then used to search for an optimal mapping, where similar pixels are clustered together. The PSO method had mixed success, but was more effective at minimising the distances between pixels within clusters, and maximising the distance between different clusters than the baseline method.

Bala *et al.* used GAs as a feature selection method to select features from images for classification using a decision tree [7]. They investigated two approaches: one using a wrapper, and one using a filter. The wrapper approach incorporates the training and evaluation of the decision tree classifier into the training of the GA, so that the performance of the GA is determined by the performance of the decision tree when using the features selected by the GA. The filter approach is simpler, in that the GA is first trained with a fitness function that evaluates the features selected by the GA in terms of entropy and cost. The trained GA is then used to train a decision tree classifier, as a separate process. They found the wrapper approach was able to give better performance, as the training of the GA was much more

effective when its fitness could be directly linked to the performance of the decision tree classifier. Using EC techniques for feature selection is a common way to improve classification performance; the use of GP for this purpose is discussed in the next subsection.

Combinations of EC techniques with other methods have also been shown to be effective at image classification [18, 30].

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* [11]. 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 an SVM, Naive Bayes [11] or Adaptive Boosting (AdaBoost) [46]. This approach is used as learning algorithms traditionally require fixed length features as input; an SVM cannot train effectively when a varying number of features is used. Farquhar *et al.* [16] proposed improvements to overcome this limitation which allowed for improved performance as training could be performed directly on input features without any information being lost.

2.6.3 Genetic Programming for Feature Selection and Construction

Feature selection is the process of selecting a subset of a feature set in order to build more effective learning models [27]. Reducing the number of features used to train a model reduces the search space for a model, which can lead to a performance improvement. Feature construction creates new, higher-level features, generally by combining multiple existing features [29]. Constructed features generally better describe an instance than a single existing feature, reducing the number of features required, which reduces the complexity and hence allows for a potential performance boost.

GP has been applied extensively to both feature selection and construction tasks [15], with particular success in feature construction due to its tree-structure which allows features to be combined using a range of functions in order 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 constructed feature. One such approach used GP as a feature construction method where the root of the tree was used by an SVM for image classification [47]. This approach created GP trees which used a large range of functions to construct features within the tree that operated on the original image. 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 achieve 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 simple compared to the SURF and HoG algorithms, which produce higher-level features.

The GPMFC algorithm uses GP to construct multiple high-level features, where one feature is constructed for each class label [40]. This algorithm uses a filter approach, where GP is used to construct features and then a classifier is trained to use the features for classification as a separate process. The GP design uses the standard arithmetic functions as function nodes, and the original features as terminal nodes. The GP process is run once for each class label, and the best program produced for each run is saved. This produces multiple GP trees, each of which produce a single feature. The fitness function evaluates how well a GP tree can separate the instances for the given class from the instances of all the other undesired classes (i.e. the purity of the class interval). Once this training process is complete, a classification algorithm is then trained which classifies an instance based on the set

of constructed features produced by the multiple GP trees. Applying GMPFC to a range of classifiers showed a general improvement compared to when the classifiers used all of the original features. The GMPFC approach is interesting in that it creates multiple independent GP trees, each of which produces a single high-level feature. Many GP approaches pick only a single tree, and hence only use a single constructed feature. By using multiple trees, the authors were better able to perform multi-class classification as the classifiers could consider multiple features in order to better distinguish classes.

2.7 Summary

This section provided an overview of the relevant literature for the ideas used in this report. A number of limitations of existing work were discussed, which provide further motivation for the work done in this report.

The use of GP for image classification was discussed, and it was found that GP was most commonly used for either feature construction using low-level features, or for classification using relatively simple features. While other machine learning algorithms have been used for classification using features produced by SURF or HoG, GP has not been applied in the same way. The use of STGP for this purpose has the potential to give good results, as advanced functions can be designed that operate specifically on these features in a way that other classifiers (e.g. SVMs) cannot.

It was also shown that the features constructed from images with GP are often quite simple (e.g. pixel statistics), which may limit the performance that can be achieved. More advanced features can be automatically generated using GP by designing functions that are based on popular feature extraction algorithms such as HoG. These functions can be optimised to a specific dataset during the evolutionary process, producing a program that is more specific and accurate than the traditional HoG approaches. The use of region selection [2] as part of a GP tree showed good results, but again extracted relatively simple features from the selected regions. Using region selection and more advanced feature extraction functions within the same GP tree has the potential to further improve performance.

Chapter 3

The Hybrid GP-SURF Approach

3.1 Introduction

This chapter details the design, evaluation and analysis of the first objective of this project. New GP methods are combined with the existing SURF algorithm for image classification. These new methods are said to use a Hybrid GP-SURF approach. The new methods are tested on a variety of datasets from different domains, and compared to existing baseline methods. Some GP programs with good performance are analysed to investigate how they are able to work well.

3.2 Chapter Goals

While the SURF algorithm has been used extensively in the literature for object detection, its use with GP techniques is limited. This chapter aims to investigate how SURF can be used with GP for image classification. This will be achieved with the following objectives:

1. Using GP for classification with features extracted from SURF as inputs. GP has been used extensively for image classification, and has shown promising performance using low-level image features. It follows that GP may be more effective when using higher-level features such as the keypoints produced by SURF. GP is a flexible technique in that it can use a variety of functions and terminals to give good performance. It is hoped that novel functions can be applied to SURF keypoints in order to increase the accuracy of image classification.

2. Using GP to select regions for keypoint extraction with the SURF algorithm. The normal SURF algorithm extracts keypoints in a global manner, i.e. by considering the entirety of an image within one run of the algorithm. While this is a good general approach, some image datasets do not have useful keypoints throughout the whole image. For example, a dataset of human facial expressions has the most useful keypoints around the nose, eyes and mouth areas. The hair or ears do not generally give much information about a person's expression, and so would produce noisy keypoints that may decrease the classification accuracy. It is possible to address this limitation of SURF by using sub-regions of an image for keypoint extraction. It is often unclear which regions contain the most useful features, so it would be useful to use GP to select useful regions. It is hoped this will allow more relevant and useful keypoints to be selected. An existing classifier (e.g. an SVM) will then be used to classify the features extracted from these keypoints, to give a classification for the image.

3. Evaluating these new approaches by comparing their performance against other commonly used classifiers on a range of datasets.

4. Analysing and interpreting the program trees of some good individuals to understand how they can perform well. Being able to understand and validate a program would increase the confidence in the new approaches.

3.3 Method

The design of the two approaches discussed above are described in turn in the following subsections.

3.3.1 GP for Classifying Keypoints

This subsection discusses the design of the method for achieving the first goal, where SURF keypoints are directly used as inputs to a GP tree for classification. A variety of different approaches were tried, two of which will be discussed: the approach with the best performance, and an earlier approach which was overly complex, giving poor results.

(1) Generating SURF Keypoints

The SURF algorithm will generate varying numbers of keypoints based on how many points of interest an image has. For example, one dataset used in this work produces between nine and 53 keypoints depending on the image used. This presents a problem for many classifiers which expect the same number of features for each instance. To address this, the algorithm in Figure 3.1 was developed. This algorithm works by altering the Hessian threshold which SURF uses to determine how large the output from the Hessian filter must be to choose a point as an interest point. The algorithm alters this threshold using a binary search until it retrieves the number of keypoints that is required. Using a fixed number of keypoints allows more effective training to occur as a solution can perform consistently across a dataset as there will be no cases where there are missing or extra keypoints.

```
keypoints ←FINDKKEYPOINTS(image,0,1,k)

function FINDKKEYPOINTS(image, lowerBound, upperBound,k)
    threshold ← (upperBound − lowerBound)/2 + lowerBound
    keypoints ← SURF(threshold)
    if |keypoints| = k then
        return keypoints
    else if |keypoints| > k then
        lowerBound ← threshold
    else
        upperBound ← threshold
    end if
    return FINDKKEYPOINTS(image,lowerBound,upperBound,k)
end function
```

Figure 3.1: Algorithm to select k best keypoints.

It is also useful to order the keypoints by their strength, so that a consistent method of ordering is maintained. This is achieved by using the algorithm in Figure 3.1 in an iterative way, as shown in Figure 3.2. This algorithm finds the best n keypoints in turn, adding the newest keypoint (i.e. the next strongest) to the ordered list after each iteration.

While these algorithms will correctly find the best n keypoints if n keypoints are available, they are somewhat inefficient when applied many times, such as in the GP training process where they become quite a large bottleneck. As such, a caching system was used so that the same set of keypoints is not computed for the same input image and number of keypoints more than once. The SURF library used[1] does not provide any measure of a keypoint's strength, and so it was initially thought that these algorithms were the only way to select the n best keypoints. However, upon deeper inspection of the SURF library, it became evident that the library could be modified to include a keypoint's strength as the difference between its output from the Hessian filter and the Hessian threshold used. This modification would then allow a much simpler, linear time algorithm to be used, which retrieves a large number of keypoints, orders them, and chooses however many are required. This modification would greatly reduce the computational time needed, but would not change any of the results that are provided in this report. As such, this modification was not implemented, but should be taken into account for any future work. The one situation which this modification works but these algorithms may not is where there are multiple keypoints with the same strength; in this case, it is possible there may be no threshold that the algorithm can find to give an exact number of keypoints. This is a rare occurrence, as the output of the Hessian filter produces a value with a high precision. As such, this problem did not arise for the datasets used in this work. Again, the modification to the SURF library should be made if any future experiments are run in order to prevent this issue.

```
function FINDNORDEREDKEYPOINTS(image, n)
    keypoints ← {}
    for k ← 0, n do
        kKeypoints ← FINDKKEYPOINTS(image, 0, 1, k)
        add (kKeypoints \ keypoints) to keypoints
    end for
    return keypoints
end function
```

Figure 3.2: Algorithm to select k best keypoints ordered by strength.

(2) The New Functions (NF) Approach

The first approach that was developed introduced a range of novel terminal and function operators. The functions were tailored specifically to operate on features produced by SURF, in the hope that this would increase performance by allowing the creation of more complex programs. The terminal and function sets used can be seen in Tables 3.1 and 3.2 respectively, and the data-types used are as follows:

- image (original image) as a 2D array of pixel values (between 0 and 255).
- surfKPVector (a single keypoint descriptor: a vector of 64 double values).

[1]JOpenSurf: Available at: `https://code.google.com/p/jopensurf/`

- surfVectors (a list of k surfKPVector – the best k keypoints found by SURF in an image).
- listIndex (integer in [0, k] where k is the number of keypoints per image).
- surfIndex (integer in [0, 63]. There are 64 values in an a SURF keypoint descriptor).

In addition, there is the double data type which uses set-based typing. Functions which have a double input can have a child which outputs either a freeDouble or terminalDouble. This provides more flexibility in crossover and mutation while still allowing functions to define a particular type of double (e.g. terminalDouble) when required. A freeDouble can be any valid double value, whereas a terminalDouble is constrained to be in the interval [0,1].

Table 3.1: The terminal set for the NF approach.

Name	Output	Comments
Constant Double	terminalDouble	Randomly generated.
Image	image	The image input.
SURF Index	surfIndex	For the vector function, randomly generated.
Constant Integer	listIndex	Randomly generated in [0,k].

Table 3.2: The function set for the NF approach.

Name	Inputs	Output	Comments
Arithmetic Operators	double×2	freeDouble	$+, -, \times$, protected$\div$, $max, min, \lvert+\rvert$ and $\lvert-\rvert$
If	double×3	freeDouble	if input1 is +ve, input2, otherwise input3.
Vector Builder	double×64	surfKPVector	Allows program to "remember" vectors.
Surf	image	surfVectors	Runs SURF on an image and gives list of descriptors (vectors).
Nth Vector	surfVectors, listIndex	surfKPVector	Returns a vector from the vector list (surfVectors[listIndex])
Descriptor value	surfKPVector, surfIndex	freeDouble	Gives surfKPVector[surfIndex] (value at index surfIndex).
Compare vectors	surfKPVector×2	freeDouble	Gives the Euclidean distance between two vectors.
Nth Closest Vector	surfKPVector, surfVectors, listIndex	surfKPVector	Gives the (n+1)th closest vector (by Euclidean distance) in the list to the supplied vector.

Classification of a given image is done by applying a threshold of 0 to the output of the GP tree; a negative output indicates a negative class and a non-negative output indicates a positive class.

This approach performed much more poorly than expected, likely due to the larger search space created by including several new functions which required a range of different inputs. For example, the *Vector Builder* function requires 64 inputs, which gives a large number of degrees of freedom that needs to be optimised during the evolutionary process. During evolution, GP may choose to randomly mutate one of these inputs, which is unlikely to significantly change a program. This is discussed further in the results (Section 3.5.2). The next approach seeks to remedy this, by using a simpler design and voting system which has a much lower complexity.

(3) The Voting Approach

This approach uses a simpler function and terminal set as well as a voting method for classify instances. Each SURF keypoint for an instance is considered separately. The feature vector from each keypoint is fed into the GP tree and produces one vote (positive or negative) for a class. The classification of the instance is then the most popular vote across all keypoints for that instance. A high-level view of this process is shown in Figure 3.3. As each keypoint is considered independently, the GP program must only train on 64 features instead of $k \times 64$ features. This factor and the smaller function and terminal sets reduce the search space, allowing GP to perform more competitively than in the NF approach.

Only a single terminal is used, which returns a value in the SURF keypoint. As there are 64 features in a keypoint, this terminal gives a randomly generated index in the interval $[0, 63]$. The function set is the eight arithmetic operators used previously (see Table 3.2).

While a voting approach removes the need to produce exactly k keypoints, the best performance will not necessarily be achieved by allowing any number of keypoints to be produced. If a small number of keypoints is produced using the default threshold, the voting approach will likely be inaccurate as classification is being performed using only a few features. As such, the number of keypoints is still fixed when using voting. Results are presented for a range of different numbers of keypoints, such that the performance of the voting approach can be evaluated with a range of features available. If the performance is consistent across different numbers of keypoints, the confidence in the robustness of the method will be improved, as achieving good performance with the voting approach will not depend on finding the optimal number of keypoints to use.

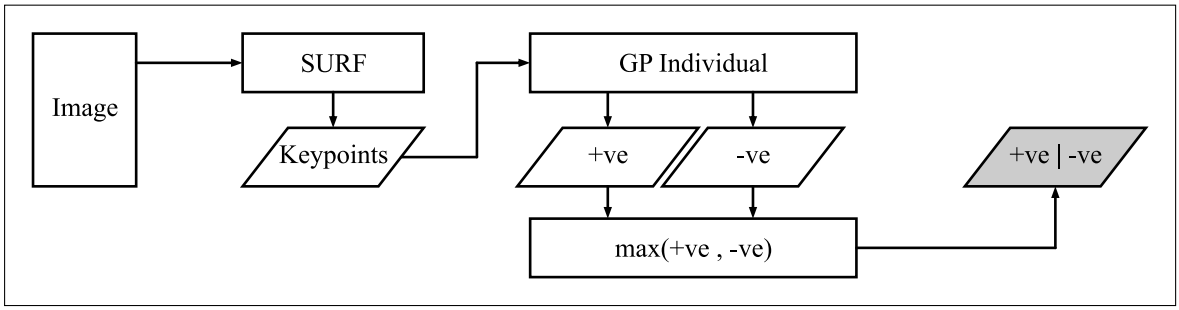


Figure 3.3: Voting approach process.

3.3.2 GP for Region Selection

This approach uses GP to select regions of an image and then applies SURF to these regions. A GP tree will output a list of regions, each of which is a rectangular section of an image to be used for keypoint extraction. A region is created in a GP tree using the *rectangle* function, that takes four arguments which define the dimensions and location of the region. The *aggregate* function allows multiple regions to be included in a list, so that a dynamic number of regions can be output by a given GP tree. Allowing a dynamic number of regions of differing size allows GP to select the most important regions in an image based on the dataset it is applied to. For example, a simpler dataset may require only one large region (i.e. similar to standard SURF), whereas a complex dataset may be classified with higher accuracy by selecting several smaller regions in different parts of the image. It is hoped that this flexibility will allow GP to select important regions which can be used to increase classification accuracy.

The function set (as described above) can be seen in Table 3.4. The terminal set can be seen in Table 3.3. The terminal set includes two terminals (*xInt* and *yInt*) which provide the location and size of regions produced by the rectangle function as well as a third terminal (*empty*) which returns an empty list with no regions. This terminal was required to overcome a limitation of ECJ [32] (the GP toolkit used for this work) whereby a tree would be unable to terminate correctly at a specified depth due to repeatedly picking aggregate nodes as children to parent aggregate nodes. An example of a GP individual utilising these functions and terminals is shown in Figure 3.4. This example will produce up to three valid regions; the empty nodes that are present are ignored. By using the empty nodes, the tree was able to terminate at a depth of four – it would otherwise need at least one additional layer to provide additional rectangle nodes. Note that the tree has no inputs that are dependent on an instance; instances are classified as part of post-processing, which allows the list of regions to be pre-computed.

Table 3.3: The terminal set for the regions approach.

Name	Output	Comments
xInt	xInt	Randomly generated in [0, minWidth] where minWidth is the minimum width of any image in the training set.
yInt	yInt	Randomly generated in [0, minHeight] where minHeight is the minimum height of any image in the training set.
empty	regionList	Empty list with no regions. Allows GP tree to terminate correctly.

Table 3.4: The function set for the regions approach.

Name	Inputs	Output	Comments
rectangle	xInt $\times$ 2, yInt $\times$ 2	regionList	Gives a list with a single rectangle region starting at(xInt1, yInt1) and with width xInt2 and height yInt2.
aggregate	regionList $\times$ 2	regionList	Combines two regionLists into one. Allows for multiple regions to be the output of a tree.

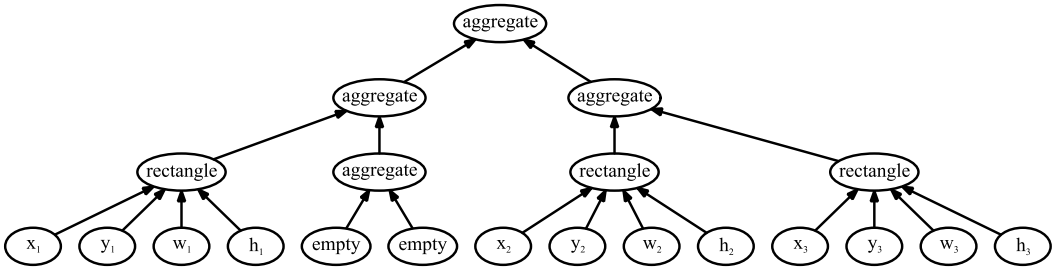


Figure 3.4: An example of the general GP program structure for the regions approach.

Classification of an instance (image) is performed using an SVM which operates on the keypoints descriptors extracted using SURF from the regions selected by the GP tree. An SVM was used as it produced the best results out of the single-source classifiers in the baseline results. An image is classified using the following steps:

1. Evaluate the GP tree to give a list of regions.
2. For each region, perform SURF on the region of that image, producing k keypoints (in the same manner as previous approaches).

3. Feed each keypoint into a trained SVM which classifies each keypoint independently as a positive vote or a negative vote.
4. Classify the image as the class with the highest number of votes.

An overview of this process is shown in Figure 3.5. The keypoints for each region must be computed separately, as SURF will operate differently on a region if it is considered separately from the whole image. This approach aims to only use sub-regions of an image rather than the whole image for feature extraction, so it is not possible to perform SURF on the whole image and then select only the keypoints that fall within each region.

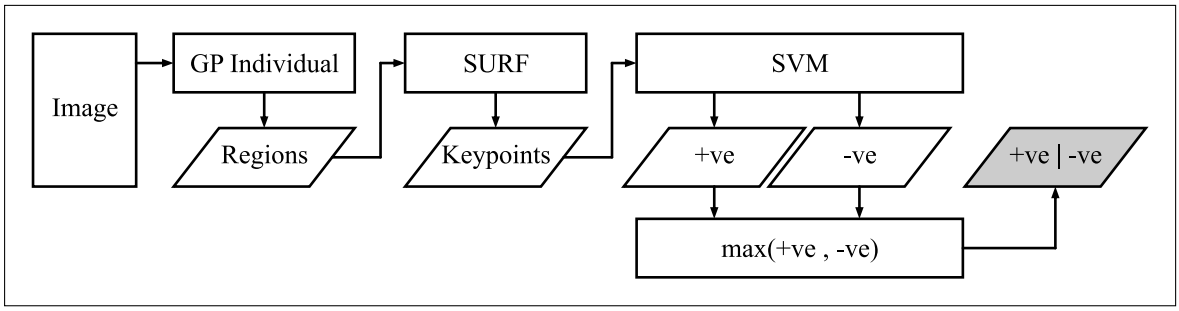


Figure 3.5: Region approach process.

Invalid regions (outside image dimensions) are ignored, and so play no part in the classification process. Similarly, regions which do not produce k keypoints before 100 iterations of the algorithm shown in Figure 3.1 are ignored as it is unlikely the correct number of keypoints will be produced with additional iterations. Regions may not produce the correct number of keypoints due to a lack of information in the region (e.g. homogeneous regions), or in the rare case that this algorithm cannot find a valid threshold due to two keypoints having the same strength (as discussed in Section 3.3.1). In either case, the region produces no keypoints and so does not affect the classification result; it will also likely be removed from the program as evolution continues. Limiting the number of iterations allows the training time to be minimised.

During training, an individual is trained by feeding all keypoint descriptors of all images into an SVM with each descriptor being labelled with the class of the image it came from. This ensures the SVM trains on each keypoint independently which is important for generalising performance as a voting approach is being used. The training performance of an individual is evaluated by finding the average accuracy across the training images when using the classification process as discussed above.

3.3.3 Fitness Function

The fitness function is used to evaluate the performance of a given GP program on a set of images. This is used during training to select the best programs to evolve, and during testing to compare the performance of the best GP program to the other classifiers. Picking a good fitness function is essential in achieving good results with GP as it determines the criterion that the evolutionary process is aiming to optimise. In this work, 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 for all the GP methods in this work is just simple accuracy which is formally defined as:

$$fitness = \frac{N_{\text{correct}}}{N_{\text{total}}} \tag{3.1}$$

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

3.4 Experiment Design

3.4.1 Datasets

Three datasets were used to evaluate the methods (NF, voting, and regions) proposed in this chapter. 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.

The two classes of the first dataset are drawn from a popular and widely used dataset in computer vision called the *Columbia Object Image Library*[2] (COIL-20) [39]. 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 then rotated through 360°, with an image captured every 5° of rotation, giving 72 images per object. The images in each class 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 *cars* and *rubber ducks* classes are used for binary classification in this work.

The *Japanese Female Facial Expression*[3] (Jaffe) [33] dataset is widely used in the literature to identify different facial expressions. There are 213 images in total in this dataset that fall into seven classes: neutral, happy, sad, surprised, angry, disgust, and fear. Ten Japanese female subjects provided several images for each facial expression. The second dataset in this report uses the instances of the happy and surprised classes, as these two expressions produce notably different images, making them suitable for use in binary classification. In this work, the instances of this dataset were manually cropped to remove the image background and most of the subject's hair, leaving only the face. This prevents the classifiers from training on irrelevant features that may give unfair results. After cropping, each instance has dimensions ranging between 121 and 143 in width, and between 164 and 207 in height.

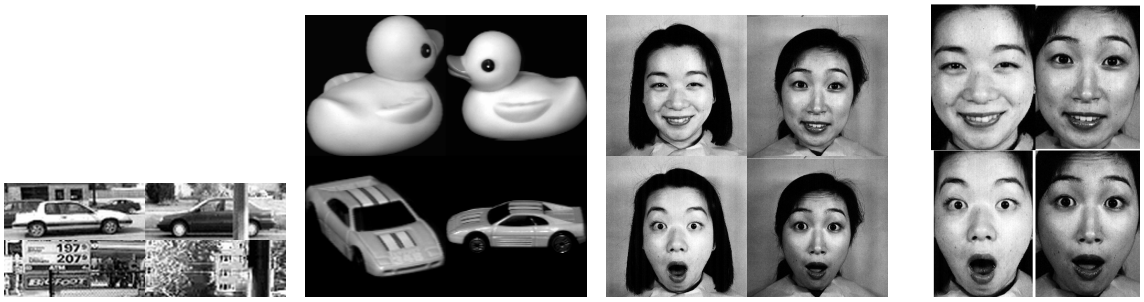
The third dataset is the *UIUC database for Car Detection*[4] (UIUC) [1] dataset. The dataset consists of $1,050$ instances of which 550 are cars and 500 are background. Each of the car instances shows the side view of the vehicle captured from the same angle and distance (giving the same scale). The instances in this dataset are 100×40 pixels in size.

Some example images used from these datasets can be seen in Figure 3.6. This figure also shows an example of the Jaffe dataset before and after cropping.

3.4.2 Training and Test Sets

For the NF approach described in Section 3.3.1, the UIUC and COIL-20 datasets were randomly split into 20% as test and 80% as training (with even ratios of positive:negative examples). The Jaffe dataset contains only three images of each expression for each human subject, requiring a careful split to ensure that the test and training sets are both representative of the dataset. As such, the Jaffe dataset uses a manual split with 66% training and 33% testing, where one image of each expression for each subject is included in the test set.

[2] Available at: `http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php`
[3] Available at: `http://www.kasrl.org/jaffe.html`
[4] Available at: `http://cogcomp.cs.illinois.edu/Data/Car/`



(a) UIUC dataset: Cars vs background. (b) COIL-20 dataset: Ducks vs toy cars. (c) Jaffe dataset: Happy vs surprised. (d) Jaffe dataset: Happy vs surprised. Cropped.

Figure 3.6: Samples of the images in each dataset.
(Top row of each dataset is positive instances, bottom row is negative).

The n-fold cross-validation technique was used to evaluate the other approaches. The instances of the UIUC and COIL-20 datasets were randomly split into 10 folds. The Jaffe dataset was manually split into three folds for the same reason as described above. Each fold contained one happy and one surprised expression for each subject. As the NF approach did not perform well, it was decided that it would not be an effective use of time to evaluate it using cross-validation.

3.4.3 Baseline Methods

A number of baseline methods are used to show how the performance of the new approaches compare to the existing methods in the literature. The Waikato Environment for Knowledge Analysis (WEKA) [19] implementations of each of the following classifiers were used:

- Support Vector Machine (SVM): a powerful classification model that analyses data and recognises patterns using an associated learning algorithm. In this work, the *Sequential Minimal Optimisation* (SMO) algorithm is used to train an SVM.
- Decision Trees (J48): a Java implementation of the C4.5 decision trees learning algorithm.
- Naïve Bayes (NB): a simple probabilistic method which follows Bayes' theorem to build a model.
- Random Forest (RF): an ensemble method that constructs a set of decision trees, which is designed to mitigate the overfitting habit of decision trees.
- Adaptive Boosting (ABM1): an adaptive method that builds a model by improving on misclassified instances of previous models.

Each of these five classifiers [49] were evaluated using the same n-fold cross-validation scheme described in Section 3.4.2. For each classifier, an instance is evaluated by giving the classifier a list of concatenated SURF keypoint descriptors. As a descriptor contains 64 values, there will be $k \times 64$ features provided, where k 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 $[a_1, a_2, .., a_{64}, b_1, b_2, ..., b_{64}]$. As the WEKA implementation of each of these methods is deterministic, they are only run once for a given experiment run. The accuracy of a classifier is found in the same way as for the GP methods, by using Equation (3.1).

The list of keypoints is ordered by the strength of each keypoint (see Section 3.3.1 for how this is defined) 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. The idea of using voting to feed each keypoint into the GP classifier independently can also be extended to the baseline methods. For example, an SVM can be trained by supplying each keypoint of an instance to it separately, along with the class label of the instance. This approach allows the baseline classifiers to also train with better generalisation properties, as they are made to vote on each keypoint in turn. The baselines with the voting approach are also used for comparison to the new methods.

3.4.4 Parameter Settings

GP has a number of parameters which can be altered in order to optimise the evolutionary process for a given problem. The three approaches in this chapter each used slightly different settings; the settings were manually chosen to improve the performance for each approach. For all approaches, the GP method was evolved for 50 generations or until perfect training performance was obtained. The other settings for each approach are as follows:

- The NF Approach: the population size was 500 and 1% elitism, 19% mutation and 80% crossover were used. The minimum and maximum program depth were two and seven respectively.
- The Voting Approach: the population size was 1,024 and top-10 elitism, 40% mutation and 60% crossover were used. The minimum and maximum program depth were two and eight respectively.
- The Regions Approach: the population size was 1,024 and top-10 elitism, 30% mutation and 70% crossover were used. The minimum and maximum program depth were two and six respectively. This smaller maximum depth was needed so the evolutionary process was able to complete in reasonable time.

3.5 Results and Discussion

This section compares the performance of each of the three approaches to that of the baseline methods, across a range of values of k. The performance of the baselines with and without voting is also discussed.

3.5.1 Baseline Methods

The results of evaluating each of the baseline methods on the datasets with n-fold cross-validation is shown in Table 3.5. The results of applying the voting technique to the baseline methods are shown in Table 3.6.

Using the voting technique gives a large performance improvement across the baselines, particularly on the Jaffe dataset. The J48 method consistently achieves above 90% performance on the test set across the datasets when using voting compared to performance as low as 56% when voting is not used. Similarly, the Random Forest method has a performance increase of around 20% across all values of k on the Jaffe dataset. It is interesting to see that the decision-tree methods (and to a lesser extent, the other classifiers) can benefit notably from using a voting approach. J48 is perhaps the worst performing classifier without voting, but is second to only Random Forest when voting is used. It appears that decision tree methods particularly benefit from classifying each keypoint independently, perhaps as they are able to best generalise the rules they use when there is a smaller number of inputs (only 64 at a time). Voting is not commonly applied in this way when using the SURF algorithm, and so the fact that this tends to improve the results on most baseline methods is a useful result found as part of this work.

Table 3.5: Performance on the baselines. Performance is calculated as per Equation (3.1). UIUC has no results for k = 20 or 50 as SURF could not generate this many keypoints.

(a) COIL-20 dataset

	$k = 5$		$k = 10$		$k = 20$		$k = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
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

(b) Jaffe cropped dataset

	$k = 5$		$k = 10$		$k = 20$		$k = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
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

(c) UIUC dataset

	$k = 5$		$k = 10$		$k = 20$		$k = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
SVM	0.99	0.92	1.00	0.91				
J48	0.99	0.84	0.99	0.83				
NB	0.90	0.89	0.90	0.89	No results*			
RF	1.00	0.94	1.00	0.93				
ABM1	0.88	0.85	0.88	0.84				

Table 3.6: Performance on the baselines with the voting approach. Performance is calculated as per Equation (3.1). UIUC has no results for k = 20 or 50 as SURF could not generate this many keypoints.

(a) COIL-20 dataset

	$k = 5$		$k = 10$		$k = 20$		$k = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
SVM	1.00	$1.00^{-\downarrow}$	$0.99^{+\uparrow}$	0.99^{-}	$0.99^{+\uparrow}$	$0.99^{-\uparrow}$	$0.99^{+\uparrow}$	0.99^{-}
J48	1.00	0.99^{-}	1.00	$1.00^{-\downarrow}$	1.00	1.00^{-}	1.00	$1.00^{-\downarrow}$
NB	$0.99^{+\uparrow}$	0.99^{-}	$0.99^{+\uparrow}$	0.99^{-}	$0.99^{+\uparrow}$	$0.99^{-\uparrow}$	1.00	$1.00^{-\downarrow}$
RF	1.00	$1.00^{-\downarrow}$	1.00	$1.00^{-\downarrow}$	1.00	1.00^{-}	1.00	$1.00^{-\downarrow}$
ABM1	$0.98^{+\uparrow}$	$0.96^{-\uparrow}$	$0.99^{+\uparrow}$	0.99^{-}	$0.99^{+\uparrow}$	$0.96^{\uparrow}$	$0.96^{+\uparrow}$	$0.95^{+\uparrow}$

(b) Jaffe cropped dataset

	$k = 5$		$k = 10$		$k = 20$		$k = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
SVM	$0.94^{+\uparrow}$	$0.90^{-\downarrow}$	$0.89^{+\uparrow}$	$0.83^{-\uparrow}$	$0.87^{+\uparrow}$	$0.84^{-\uparrow}$	$0.87^{+\uparrow}$	$0.77^{\uparrow}$
J48	1.00^{-}	$0.92^{-\downarrow}$	1.00^{-}	$0.95^{-\downarrow}$	1.00^{-}	$0.92^{-\downarrow}$	1.00^{-}	$0.97^{-\downarrow}$
NB	$0.58^{+\uparrow}$	$0.56^{+\uparrow}$	$0.62^{+\uparrow}$	$0.59^{+\uparrow}$	$0.80^{+\uparrow}$	$0.83^{-\uparrow}$	$0.91^{+\uparrow}$	$0.87^{-\uparrow}$
RF	1.00^{-}	$0.97^{-\downarrow}$	1.00^{-}	$0.97^{-\downarrow}$	1.00^{-}	$0.98^{-\downarrow}$	1.00^{-}	$0.97^{-\downarrow}$
ABM1	$0.96^{-\uparrow}$	0.87^{-}	$0.78^{+\uparrow}$	$0.73^{\uparrow}$	$0.83^{+\uparrow}$	$0.78^{-\uparrow}$	$0.75^{+\uparrow}$	$0.71^{+\uparrow}$

(c) UIUC dataset

	$k = 5$		$k = 10$		$k = 20$		$k = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
SVM	$0.93^{-\uparrow}$	0.92^{-}	$0.90^{+\uparrow}$	$0.90^{-\uparrow}$				
J48	$1.00^{-\downarrow}$	0.92^{-}	$1.00^{-\downarrow}$	$0.94^{-\downarrow}$				
NB	$0.89^{+\uparrow}$	$0.89^{-\uparrow}$	$0.88^{+\uparrow}$	$0.89^{-\uparrow}$	No results*			
RF	$1.00^{-\downarrow}$	$0.95^{-\downarrow}$	$1.00^{-\downarrow}$	$0.96^{-\downarrow}$				
ABM1	$0.88^{+\uparrow}$	$0.87^{+\uparrow}$	$0.86^{+\uparrow}$	$0.85^{+\uparrow}$				

3.5.2 GP for Classifying Keypoints

(1) The NF Approach

The NF approach was compared to the SVM baseline method across the three datasets and for a range of different numbers of keypoints (k). The method used to select k keypoints is discussed in Section 3.3.1. The number of features available to a GP program is $k \times 64$ as each keypoint has 64 values. As GP is a stochastic method, each experiment was run 35 times with different seeds, and the average result was then computed. Results are shown in Table 3.7. These results show that the GP method has a worse performance than the SVM across all of the datasets and for all values of k. This is likely a consequence of the NF approach having a particularly large search space due to the number of different functions, some of which had many children or required other functions as children. A large search space causes the GP process to be less effective in finding the best solutions, reducing the accuracy of the final solution. The SVM method nearly always achieves 100% performance on the training set,

suggesting that there is enough information in the features to allow for very good training. However, the GP method achieves a minimum of 69% and a maximum of 90% training performance, suggesting that the evolutionary process could not consistently train good programs due to the large search space. Other baselines were not used for comparison as the SVM already performs much more accurately than this approach, making it an inefficient use of time to run experiments for other baselines.

Table 3.7: Performance on the NF approach. Performance is calculated as per Equation (3.1). UIUC has no results for k = 20 or 50 as SURF could not generate this many keypoints.

(a) COIL-20 dataset

	$k = 5$		$k = 10$		$k = 20$		$k = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
Average	0.90	0.80	0.90	0.85	0.89	0.84	0.90	0.82
Max	0.98	0.96	0.98	1.00	0.97	1.00	0.99	0.96
Std Dev	0.05	0.06	0.05	0.08	0.06	0.07	0.05	0.05
SVM	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

(b) Jaffe cropped dataset

	$k = 5$		$k = 10$		$k = 20$		$k = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
Average	0.86	0.58	0.81	0.57	0.81	0.55	0.83	0.56
Max	0.98	0.80	0.93	0.85	0.95	0.70	0.95	0.85
Std Dev	0.06	0.12	0.06	0.13	0.06	0.09	0.05	0.12
SVM	1.00	0.80	1.00	0.75	1.00	0.75	1.00	0.90

(c) UIUC dataset

	$k = 5$		$k = 10$		$k = 20$		$k = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
Average	0.70	0.67	0.69	0.66				
Max	0.85	0.81	0.77	0.75				
Std Dev	0.03	0.05	0.03	0.05	No results*			
SVM	0.99	0.93	1.00	0.91				

(2) The Voting Approach

The voting approach was compared to a range of baselines using n-fold cross validation. As before, each experiment is run 35 times and then an average across all runs is computed. 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 n folds of that run. In other words, the performance reported is the average of 35 runs, each taking the average of 10 folds, where a fold's performance is that of the individual with the best training performance. This requires 350 runs of the GP evolutionary process, which can use up to 50 generations. On each generation, each of the 1,024 individuals must be evaluated across the whole dataset; the largest dataset (UIUC) contains 1,050 images. This process requires a large amount of

Table 3.8: Performance on the voting approach. Performance is calculated as per Equation (3.1). UIUC has no results for k = 20 or 50 as SURF could not generate this many keypoints.

(a) COIL-20 dataset

	$k = 5$		$k = 10$		$k = 20$		$k = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
Average	1.00	0.95	1.00	0.96	1.00	0.96	1.00	0.97
Max	1.00	0.98	1.00	0.99	1.00	0.99	1.00	1.00
Std Dev	0.00	0.01	0.00	0.02	0.00	0.02	0.00	0.02

(b) Jaffe cropped dataset

	$k = 5$		$k = 10$		$k = 20$		$k = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
Average	0.95	0.71	0.96	0.73	0.95	0.72	0.96	0.75
Max	0.98	0.87	0.99	0.85	1.00	0.85	1.00	0.92
Std Dev	0.02	0.07	0.02	0.05	0.03	0.08	0.02	0.07

(c) UIUC dataset

	$k = 5$		$k = 10$		$k = 20$		$k = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
Average	0.91	0.88	0.91	0.87				
Max	0.93	0.90	0.92	0.90	No results*			
Std Dev	0.01	0.01	0.01	0.01				

computation time, especially when different numbers of keypoints were also tested. The results for this approach is shown in Table 3.8.

Baseline without voting: This approach is competitive when compared to the baseline approaches (Table 3.5), with only minor variations in performance across all datasets when compared to SVM, Naïve Bayes and AdaBoost M1. The GP approach performs better than J48 (decision trees) but worse than Random Forest. Random Forest works by using multiple decision trees, whereas the GP technique only uses the best individual (a single tree) that was evolved. Hence, GP can be considered to have performed well by only using a single classifier.

Baseline with voting: When voting is used on the baseline approaches (Table 3.6), the GP method performs relatively poorly. Student's t-test was used with a 95% confidence interval to evaluate the significance of the performance variations. In the table, a "+" is used to indicate when the GP method is significantly better than a given baseline; the "−" symbol indicates the GP method was significantly worse. The higher performance of the baseline classifiers when using voting appears to allow them to outperform the GP method. The GP method performs significantly better than AdaBoost M1 and Naïve Bayes on some experiment runs, but is generally performs significantly worse against the other three classifiers. The arithmetic functions that are used in the GP tree are less powerful than the hyper-plane technique used in SVM, or the use of multiple decision trees as in Random Forest. This likely makes GP less effective in comparison.

3.5.3 GP for Region Selection

The region approach was also compared to the baseline methods using n-fold cross validation. The results were averaged across 35 runs in the same manner as in the voting approach described previously. The results for this approach are shown in Table 3.9.

Table 3.9: Performance of the regions approach. Performance is calculated as per Equation (3.1).

(a) COIL-20 dataset

	$k = 5$		$k = 10$		$k = 20$		$k = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
Average	1.00	0.99	1.00	0.99	1.00	1.00	1.00	0.99
Max	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Std Dev	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01

(b) Jaffe cropped dataset

	$k = 5$		$k = 10$		$k = 20$		$k = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
Average	1.00	0.87	1.00	0.87	1.00	0.89	1.00	0.89
Max	1.00	0.95	1.00	0.94	1.00	0.95	1.00	0.95
Std Dev	0.01	0.05	0.01	0.05	0.01	0.04	0.01	0.03

(c) UIUC dataset

	$k = 5$		$k = 10$		$k = 20$		$k = 50$	
	Training	Test	Training	Test	Training	Test	Training	Test
Average	0.99	0.92	0.98	0.92				
Max	1.00	0.98	1.00	0.95	No results*			
Std Dev	0.01	0.03	0.01	0.03				

Baseline without voting: The results produced by this approach are notably better than the baseline methods (Table 3.5) on the Jaffe dataset and have nearly perfect fitness on the COIL-20 dataset (in line with the best baselines). The Jaffe results are particularly good as the previous approaches have always struggled to perform well on the test set for this dataset. The results are also consistent across different numbers of keypoints which increases the confidence in this method.

Baseline with voting: The performance of this region approach is also comparable to the baselines with voting (Table 3.6). Again, Student's *t*-test was used with a 95% confidence interval to evaluate the significance of any performance variations. In the table, a "↑" symbol is used to indicate when the GP method is significantly better than a given baseline; the "↓" symbol indicates the GP method was significantly worse. The average performance is significantly higher than that of the SVM baseline on three out of four of the Jaffe cropped runs and is much more consistent; the performance of GP ranges from 87% to 89% on Jaffe, whereas the SVM baseline ranges from 77% to 90%. As the GP method uses an SVM as a classifier, this higher consistency suggests that the use of GP has improved the results that

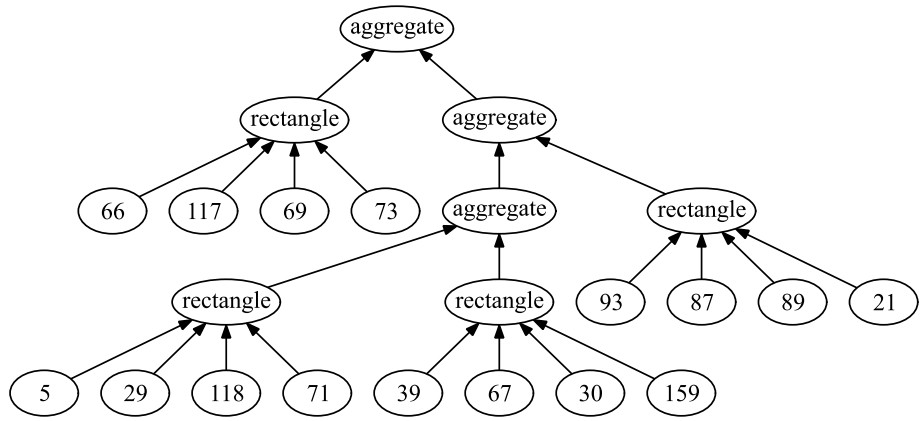
can be achieved with an SVM. The decision tree-based methods using voting (J48 and Random Forest) are significantly better than GP across the three datasets. As discussed before, Random Forest uses multiple decision trees and so has an advantage as the GP method does not use multiple GP trees. GP performs significantly better than both Naïve Bayes and AdaBoost M1, and so performs well in comparison to three of the five baselines.

3.6 Further Analysis

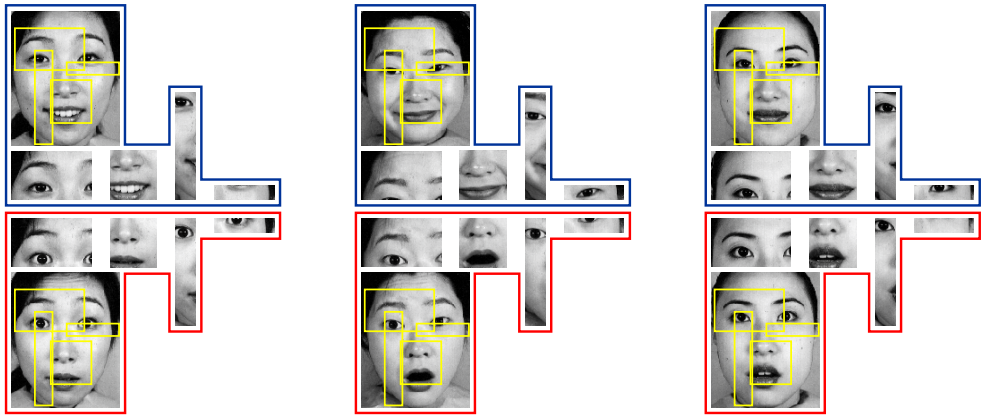
The regions approach was the most successful of the three approaches proposed in this chapter. It also produced a number of high-performing but relatively simple programs which are able to be understood by humans. The section focuses on analysing two such programs, with a particular focus on explaining how the regions selected by this approach may produce particularly good keypoints. This analysis provides an insight into why the regions approach is able to perform well.

3.6.1 Example Program 1

Figure 3.7 (a) shows the tree representation of a program evolved on the Jaffe dataset using the regions approach. This program achieved on average over 95% accuracy on the unseen data across 3-fold cross validation.



(a)



(b) (c) (d)

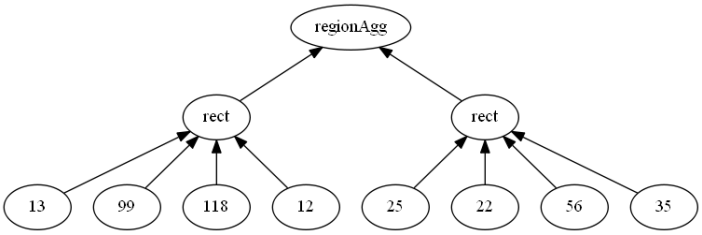
Figure 3.7: Example program 1: (a) is the tree representation of a program evolved on the Jaffe dataset and (b)–(d) are the detected regions on three samples.

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 Figure 3.7(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 bordered with blue; for the surprised class, they are bordered by 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 (contrasting) features are extracted for better classification performance.

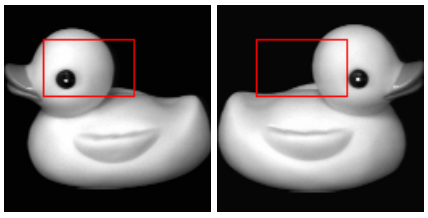
3.6.2 Example Program 2

The baseline and GP methods all achieved very good performance on the COIL-20 dataset, as it is relatively simple. However, it is still useful to analyse an evolved program which has perfect performance on this dataset in order to understand how GP is able to easily classify the images in it. Figure 3.8 shows a program with perfect accuracy which uses the regions approach. This program contains two regions, but only one is valid – the other falls outside the image boundaries and so is discarded. Hence, only a single region is used for classification.

This region is shown for two instances of each class in Figure 3.8 (b)–(e). This region is relatively small, indicating that the image can be classified with only a small amount of information. The region always contains the edge of the object in the image, suggesting that SURF is extracting keypoints corresponding to an edge which distinctly identifies the image's class. For the duck instances, the region contains the edges of the duck's head and neck. On the toy cars, the edge corresponding to the top of the cars is in the centre of the region. The plain black background gives a large contrast to the toy objects, allowing easy extraction of edge keypoints, and a high classification accuracy.

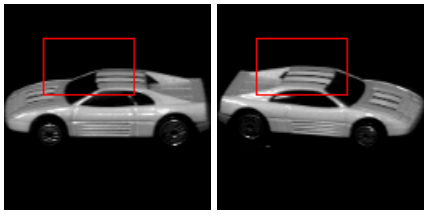


(a) GP tree



(b) Duck oriented left.

(c) Duck oriented right.



(d) Car oriented left.

(e) Car oriented right.

Figure 3.8: Example program 2: 100% training and 100% test performance on the COIL-20 dataset.

3.7 Chapter Summary

This chapter introduced two new Hybrid GP-SURF approaches which aimed to improve the performance of image classification by combining the SURF algorithm with new GP techniques. The first approach, using GP for classification of SURF keypoints, had limited success. However, the voting technique used as part of this approach proved to improve results when applied to some existing machine learning methods. The second approach, using GP for region selection, had promising performance that likely could be improved even further by using a decision-tree based classifier. The regions selected by the second approach were also analysed to give an insight into how GP was able to give good performance. Combining GP with SURF in these ways provides a novel contribution in the computer vision domain.

Chapter 4

The GP-HoG Approach

4.1 Introduction

This chapter discusses the design, evaluation and analysis of the second objective of this work. A new GP program design with novel functions inspired by the Histogram of Oriented Gradients (HoG) algorithm are proposed in order to improve the performance of image classification compared to some machine learning baselines. The performance of this GP-HoG approach is compared to the baselines and datasets used in Chapter 3. Some GP programs with good performance are analysed to investigate how GP is able to create useful programs.

4.2 Chapter Goals

The Hybrid GP-SURF approaches proposed in Chapter 3 combined SURF and GP in a two-stage manner. The feature extraction algorithm (SURF) was left unchanged. While the SURF algorithm produces good general features across a range of datasets, features that are more specific to a dataset would allow more accurate classification. Using GP to also perform feature extraction would allow features specific to a dataset to be produced. This chapter aims to develop a new approach which uses GP for region selection and classification (as in Chapter 3), but also for feature extraction. Incorporating all of these steps into a single GP tree allows for the whole classification system to be trained simultaneously.

The Two-Tier GP (2TGP) method [2] used GP to select important regions of an image, extract useful features from those regions and then performed classification using those features. A range of different region shapes (circles, rectangles, rows, columns) were used, and features were extracted from these regions using a range of aggregation operators. These operators applied statistical techniques (including the mean, standard deviation, minimum, and maximum) across the pixels in a region to produce a feature value representing a feature extracted from that region. While this approach achieved good performance, the feature extraction functions used produced relatively simple features.

The HoG algorithm is an example of a feature extraction algorithm which produces higher-level features, based on the orientation of gradients in an image. Gradients of an image can give a higher amount of information about the structure of an image than pixel statistics such as mean and standard deviation can.

Combining the ideas behind the HoG algorithm with the region selection approach has the potential to give good classification performance by using the evolutionary process to automatically select good regions, extract useful high-level features and then perform classification. The goals of this chapter are as follows:

1. Develop new functions which are inspired by the HoG algorithm. These functions will allow GP to produce high-level features which have the potential to increase classification performance.

2. Combine these new functions with the region selection approach from the 2TGP method to allow GP to perform region selection, feature extraction and classification in one tree.

3. Evaluate this new approach against the baseline classifiers and datasets (as used in Chapter 3).

4. Analyse the program trees of some good individuals to understand how they are able to achieve high classification accuracy with only a few high-level features.

4.3 Method

This new approach uses a combination of terminals and functions from the 2TGP method and some novel terminals and functions inspired by the HoG algorithm. The full terminal and function sets can be seen in Tables 4.1 and 4.2 respectively. 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 shape, coords, and aggWindow terminals define the shape, size and location of a region. The arithmetic operators in the function set allow GP to utilise multiple extracted features for classification. The HoG, compHistograms and histogramVal functions and the histogramIndex terminal are used for feature extraction. The design of these functions are discussed in the next subsection.

The same fitness function is used as in the Hybrid GP-SURF approach. This fitness function is defined in Section 3.3.3.

Table 4.1: The terminal set for the GP-HoG approach. All terminals except histogramIndex are from [2].

Name	Output	Comments
Image	image	The image being evaluated.
Random Double	double	Random double in [0,1).
Coords	coords	Random x in $[0, minImageWidth]$, random y in $[0, minImageHeight]$.
Agg. Window	aggWindow	Random int in $[3, min(minImageWidth, minImageHeight)]$.
Shape	shape	one of rectangle, circle, row, column. Rectangle has x in $[0, minImageWidth]$ and y in $[0, minImageHeight]$.
Histogram Index	histogramIndex	Random integer in [0,7]. Used to specify the bin of a histogram to be used.

4.3.1 HoG-Inspired Functions for Feature Extraction

The most important new function is the HoG function, which is inspired by the HoG algorithm [12]. The HoG function takes the image, coords, shape, and aggWindow 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 eight bins is used, where each bin corresponds to 45° of rotation [12].

Table 4.2: The function set for the GP-HoG approach.The arithmetic operators are from [2].

Name	Inputs	Output	Comments
Arithmetic Operators	double×2	double	$+, -, \times$, protected $\div$.
HoG	image, coords, shape, aggWindow	histogram	Performs the HoG algorithm using the given inputs.
Compare Histograms	histogram ×2	double	Returns the Euclidean distance between two histograms.
Histogram Value	histogram, histogramIndex	double	Returns the double at histogram$[histogramIndex]$.

Outline of the HoG function

First, the region of the image is selected based on the inputs to the HoG function. This is done in the same manner as in 2TGP, by taking a region of an image defined by the shape (how the region is shaped), coords (the position of the region) and aggWindow (size of the region) inputs. Then, the following steps are applied:

1. Apply the $[-1, 0, 1]$ and $[-1, 0, 1]^T$ masks to each pixel in the region (i.e. convolution using the two masks). Let the result of these two masks be iX and iY respectively.
2. Find the magnitude at each pixel as $\sqrt{iX^2 + iY^2}$.
3. Find the orientation of each pixel as $\arctan \frac{iY}{iX}$. This is converted to degrees and mapped to be in the interval $[0, 360^\circ)$.
4. For each pixel:
   - Find the two bins of the histogram it lies between based on its orientation. The histogram is divided into eight bins of size 45°, so a pixel with an orientation of 80° would have its lower bin as bin two (45°), and its upper bin as bin three (90°).
   - 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.
   - For each bin, calculate *weightedMagnitude* as the pixel's magnitude multiplied by how close it is to that bin. Add *weightedMagnitude* to the bin. As the bin size is 45°, $magnitude \times \frac{(45-35)}{45}$ is added to the lower bin. The upper bin would have $magnitude \times \frac{(45-10)}{45}$ added to it, as the upper bin's orientation is closer to that of the pixel.
5. Normalise the histogram by expressing the value of each bin as a fraction of the sum across all bins.

The above method differs from the standard HoG algorithm [12] 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 algorithm, the normalisation process is only applied across 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 shaped; normally the design of blocks are fixed, e.g. as an 8×8 square. It is hoped the evolutionary process will be able to automatically find the best block sizes and shapes as individuals with the best block designs will be rewarded with a higher fitness. The crossover operator also allows useful block designs to be exchanged between individuals.

As a number of variations to the original HoG algorithm have been made, it is best to consider the function used to be inspired by the HoG algorithm, rather than a strict implementation of it within the GP approach. As the above steps show, this function outputs a feature vector (histogram) with eight bins. GP can not directly use a vector for classification, and so two additional functions were designed which construct higher-level features from a histogram. These are described below:

Functions for feature construction

The compHistograms function finds the Euclidean distance between two histograms. This produces a double value which is used as a measure of how similar 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, the regions corresponding to a car's front and back wheels will produce similar histograms. These same regions on a background image are likely to give different histograms. Hence, the distance between histograms can be used as a feature for classification.

The histogramVal function returns the value of a given bin of a histogram. This function allows GP to select important orientations which have different magnitudes depending on the image class. On the Jaffe dataset, the edges of the mouth have a different gradient when the subject is happy compared to when they are surprised. The magnitude of a given bin can be used as a feature for distinguishing two classes.

4.3.2 The Classification Process

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 double value output. A threshold is then applied to this value. A negative value gives a negative classification; 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.

4.4 Experiment Design

The new GP-HoG and the 2TGP method were both applied to the datasets discussed in Section 3.4.1. For the 2TGP method, 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 two and 10 respectively. The evolutionary process was run for 50 generations or until perfect training performance was obtained. The GP-HoG method 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 the 2TGP approach. A third set of experiments were also run on the GP-HoG approach with a larger population size of 10,000. Increasing the population size can allow for greater training performance (and hopefully test performance), as having more individuals increases the chance of finding an even better

solution. As before, all three experiments used n-fold cross-validation to increase the confidence in the validity of the results. The details of how these folds were split are discussed in Section 3.4.2. As before, the average of 35 runs with different seeds are reported.

4.5 Results and Discussion

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

4.5.1 Compared to the 2TGP Approach

The results for the 2TGP approach are shown in Table 4.3 and the results for the new GP-HoG approach are shown in Tables 4.4 and 4.5. 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 4.3 indicates that GP-HoG with a 1,024 population size was significantly better than the 2TGP approach; A "$-$" indicates it was significantly worse. Similarly, a "$\uparrow$" indicates GP-HoG with a 10,000 population size performed significantly better than 2TGP, and a "$\downarrow$" indicates it was significantly worse.

The GP-HoG approach performs significantly better on the Jaffe and UIUC datasets (the two difficult datasets) while achieving slightly worse test performance on the COIL-20 dataset. The performance on the Jaffe dataset is improved even further when the population size of 10,000 is used (Table 4.5), achieving 82% accuracy on the test data compared to 71% accuracy with the 2TGP approach. These improvements show that using more advanced feature extraction techniques can improve the classification performance by producing features that more precisely differentiate images of different classes.

Training Time

The average training time is increased notably using the new approach, with approximately $3\times$ more computation required on the Jaffe and UIUC datasets when the same population size (1,024) is used. The training time on COIL-20 is increased significantly more, but is still quicker than for the other two datasets. This is 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 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. The training time is increased even further with a population size of 10,000. It is around $5\times$ slower on the COIL-20 and Jaffe datasets; it would be expected to be $10\times$ slower as the population size is increased tenfold, but as nearly all runs on these datasets achieve perfect training performance with the increased population size, the evolutionary process nearly always stops before the 50 generation limit, reducing training time. On the UIUC dataset, the average training time increases nearly elevenfold when the population size is increased. Training never finishes early on this dataset (the maximum training performance is 98%), and so the training time is much slower. Even with the utilisation of multithreading, the 600 hours of CPU time 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 once trained. If a lower training time is desired, the population size can be decreased with an associated decrease in performance.

Table 4.3: Performance of the 2TGP method proposed in [2]. Performance is calculated as per Equation (3.1).

	COIL-20		Jaffe cropped		UIUC	
	Training	Test	Training	Test	Training	Test
Average	1.00	0.99 $^{-\downarrow}$	0.95 $^{+\uparrow}$	0.71 $^{+\uparrow}$	0.94 $^{+\uparrow}$	0.92 $^{+\uparrow}$
Max	1.00	1.00	0.98	0.81	0.96	0.94
Std. Dev.	0.00	0.01	0.02	0.05	0.01	0.01
Average Training Time (H:M:S)	00:05:49		01:35:54		16:22:39	

Table 4.4: Performance of the GP-HoG method (1,024 population size). Performance is calculated as per Equation (3.1).

	COIL-20		Jaffe cropped		UIUC	
	Training	Test	Training	Test	Training	Test
Average	1.00	0.97	0.97	0.76	0.95	0.93
Max	1.00	1.00	0.99	0.92	0.96	0.95
Std. Dev.	0.00	0.02	0.02	0.07	0.01	0.01
Average Training Time (H:M:S)	03:24:26		04:36:24		52:15:25	

Table 4.5: Performance of the GP-HoG method (10,000 population size).Performance is calculated as per Equation (3.1).

	COIL-20		Jaffe cropped		UIUC	
	Training	Test	Training	Test	Training	Test
Average	1.00	0.98	1.00	0.82	0.97	0.95
Max	1.00	1.00	1.00	0.94	0.98	0.95
Std. Dev.	0.00	0.01	0.00	0.06	0.00	0.01
Average Training Time (H:M:S)	16:57:28		21:39:55		603:06:38	

4.5.2 Compared to the Baselines

Baselines without voting

When comparing the GP-HoG approach to the baselines using SURF, the new approach has comparable performance on the COIL-20 dataset and improved performance on the Jaffe and UIUC datasets when voting is not used on the baselines (see Table 3.5). 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.

Baselines with voting

When voting is used on the baselines, the GP-HoG method generally performs worse on the Jaffe dataset, but is comparable to the performance of AdaBoost M1 and Naïve Bayes. By using a voting approach, the baseline methods are able to give very good performance as they consider multiple keypoints independently – if a few keypoints are misclassified, it will not affect the classification result. The baseline classifiers are also use more sophisticated techniques than GP (which uses arithmetic building blocks for classification) and so are able to use the larger feature vectors that SURF offers to give more accurate performance on

the Jaffe dataset. The GP-HoG method is competitive on the easier COIL-20 dataset, and performs as well as the best baseline (Random Forest) on the UIUC dataset when a 10,000 population size is used. This shows the potential of the GP-HoG method, as it is able to give good classification performance by extracting only a few useful features. Chapter 5 will show how using only two or three features produced by the GP-HoG method can give over 90% classification accuracy on the Jaffe and UIUC datasets. The GP-HoG approach also has the potential to be improved further by incorporating other useful feature extraction algorithms into the function set. Using only a few features also reduces the computation time in comparison to the SURF when a trained individual is used for classification.

4.6 Further Analysis

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

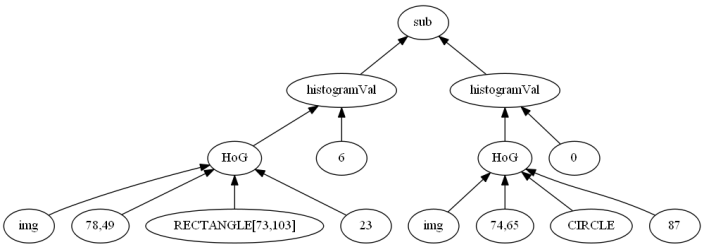
4.6.1 Example Program 3

An example of an evolved program with high performance on the Jaffe dataset is shown in Figure 4.1. It is an interesting program to analyse, as it is very simple: it consists of two HoG operators, and a subtraction operator. The left side of the tree applies the HoG operator to a rectangular region which corresponds to the right side of the face, including the eye, cheek, and part of the nose, lip, and eyebrow area. 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, producing a different orientation of the gradient than when they are surprised. When they are surprised, the mouth is lowered, creating a right-angle between the chin and where the subject's ear would be. This gives a larger orientation for this edge gradient than the smaller angle for the same edge in the happy expression, changing the histogram that is produced. This HoG operator produces a histogram, and the value of the bin corresponding to the $270^\circ - 314^\circ$ orientation range is then selected by the histogramVal function.

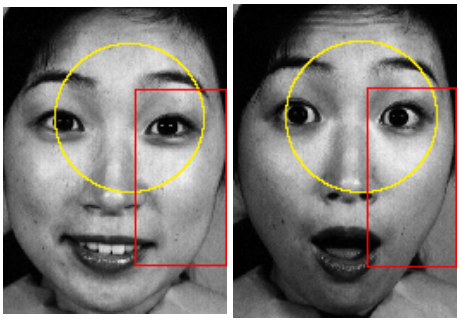
On the right side of the tree, the HoG operator is applied to a circular region of the image, which corresponds to the upper nose, eyes, and eyebrow area. The subject's nose appears much narrower in the surprised expression, due to her mouth being open. There is also more of the nose included in the surprised image, which introduces an additional edge gradient. The eyes are also obviously different when surprised, as they appear wider and have more white showing. All of these differences affect the histogram that is produced, allowing GP to extract features that distinguish these two expressions. The value of the bin corresponding to the $0 - 44^\circ$ range is chosen by the histogramVal node on the right hand side of the tree. The root of the tree then outputs the difference between the values of the bin from the left and right sides of the tree. This simple program achieves 98% training and 95% test performance on one of the folds of this dataset.

4.6.2 Example Program 4

Another program with very good performance (95% training and 100% test accuracy) on the Jaffe dataset is shown in Figure 4.2. This program is similar to the previous one 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 quite a different appearance between the happy and surprised expressions – when the subject is surprised, a larger amount of their eye is visible. A greater amount of



(a) GP tree.

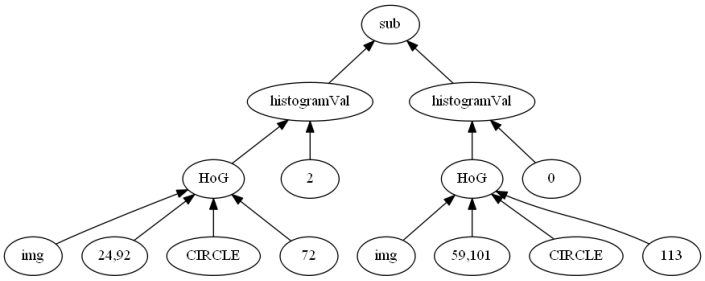


(b) Happy (+ve) expression.

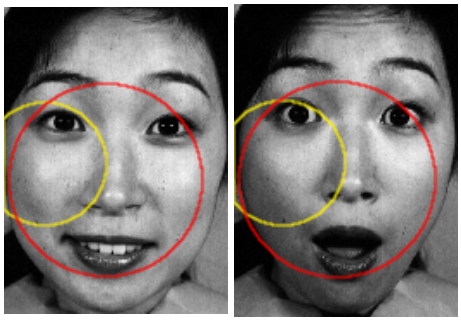
(c) Surprised (-ve) expression.

Figure 4.1: Example program 3: 98% training and 95% test performance on Jaffe dataset.

the nostril is also included in the yellow circle when surprised. These differences produce a different histogram of gradients. The surprised expression has a more significant gradient near the nostril than the happy one, as more black pixels are shown when the subject is surprised. While the red circle is somewhat 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 appears much darker in the surprised expression, and so has a lower amount of gradient than 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 red circle can be used to distinguish between happy and surprised expressions.



(a) GP tree.



(b) Happy (+ve) expression.

(c) Surprised (-ve) expression.

Figure 4.2: Example program 4: 95% training and 100% test performance on Jaffe dataset.

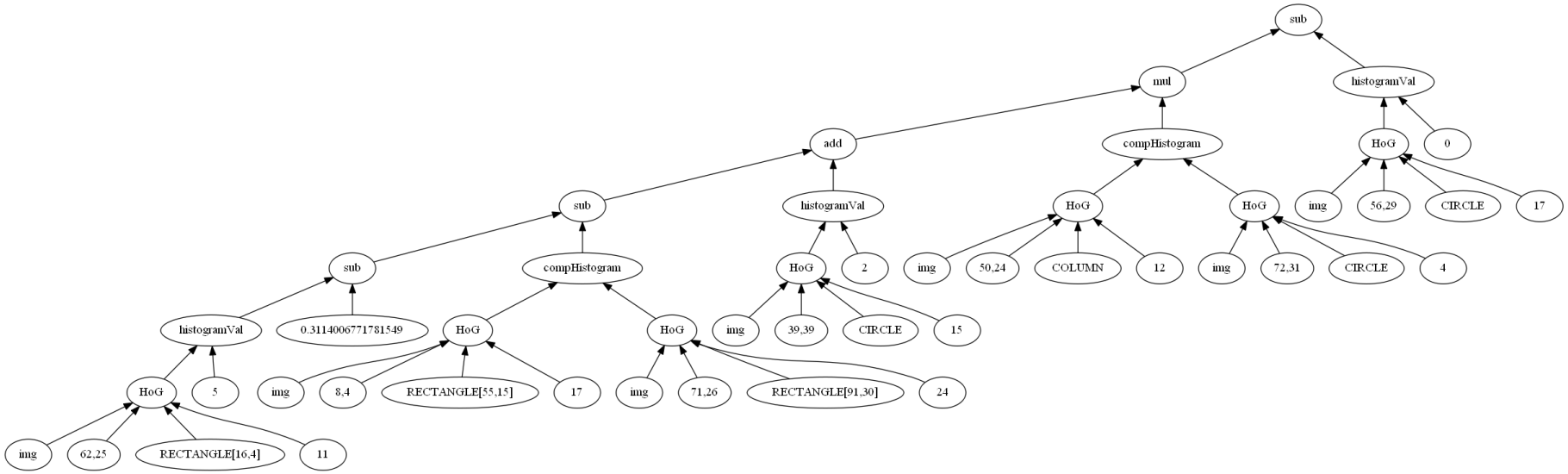
4.6.3 Example Program 5

When GP-HoG is applied to the UIUC dataset, much larger programs are required to obtain a high classification accuracy. Unlike the Jaffe and COIL-20 datasets, GP-HoG is unable to obtain perfect training performance on UIUC, and so large programs are needed as part of the evolutionary process in order to maximise the training performance. The example program in Figure 4.3 is one of the simpler programs produced that performs well on this dataset, with a tree depth of eight. The regions used by this program are shown in Figure 4.3 (a) and (b). While many regions are used in the tree, all regions tend to be relatively small, and so identify particular aspects of the image that are useful for classification. Several of the regions enclose specific parts of the car in (a), such as the front wheel, wheel arch and

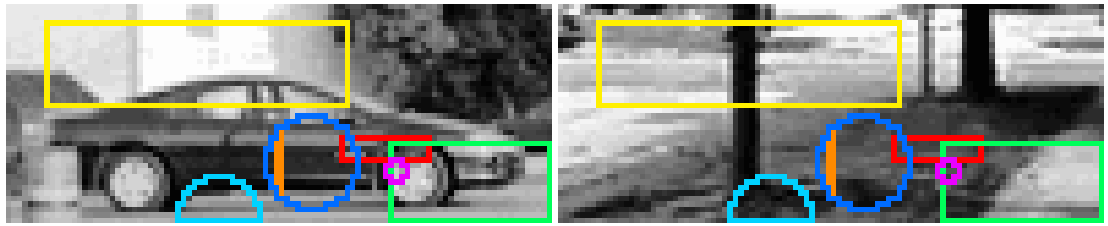
parts of the front door. The car wheel in particular is likely to have a distinctive gradient pattern, as it has a circular tyre surrounding the hubcap. This gives a circular edge which has a consistently changing orientation and a large gradient magnitude (as the tyre is black and the hubcap is grey), which produces a histogram with a similar magnitude in each bin. The same area where the car wheel is in the background image contains a straight edge, which would produce a histogram with a spike in one bin. This difference in histograms can help the GP tree to distinguish these two classes.

4.7 Chapter Summary

A new GP method was proposed in this chapter which combined functions inspired by the HoG algorithm with the region selection concept proposed in the 2TGP method. Performance evaluation showed good performance using the proposed method, especially when a large population size was used. On the most difficult dataset (Jaffe), 82% testing performance was achieved, an increase of 11% over the 2TGP approach. 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 (e.g. with a depth of 5). The adaptation of the HoG algorithm for use as a GP function was shown to be an effective method of performing high-level feature extraction directly within a GP tree.



(a) GP tree.



(b) Car (+ve) image.

(c) Background (-ve) image.

Figure 4.3: Example program 5: 96% training and 97% test performance on UIUC dataset.

Chapter 5

Analysis of GP-HoG for Feature Extraction and Construction

The evaluation of the methods proposed in this work so far has focused only on the classification accuracy across the datasets. However, an important characteristic of the GP-HoG method is its ability to perform all of region selection, feature extraction, and classification simultaneously. It performs feature extraction on images by selecting regions and applying a feature extraction algorithm to produce features. For example, it uses histograms produced by the HoG function for classification, by using the histogram as a feature vector. The GP-HoG approach also performs feature construction, whereby it chooses individual values from a histogram using the histogramVal function or compares histograms using the compHistogram function, producing higher-level features.

5.1 Chapter Goals

This chapter investigates the effectiveness of the GP-HoG approach as a feature extractor and constructor by using features produced by some good GP trees as inputs to the baseline classifiers. Performance will be evaluated by comparing classification accuracy to that of the baseline classifiers using SURF keypoints, as well as to the original GP trees.

5.2 GP-HoG for Feature Extraction

The HoG function produces histograms with eight bins, giving eight features that can be used for classification. While the GP method selects individual bin values, other classifiers may perform best when they have access to the whole histogram due to the additional amount of information available. The GP tree is used here for feature extraction by concatenating each of the histograms used in the program to give a feature vector of size $h \times 8$, where h is the number of histograms in the program. Hence, the GP tree is used to select useful regions for feature extraction with the HoG function.

This approach was applied to each of the three example GP-HoG programs discussed previously (Figures 4.1–4.3) by feeding the feature vector into each of the five baseline classifiers. The performance of each classifier was found by applying the same n-fold cross-validation approach used previously, and is compared to the performance of the original GP tree (also using n-fold cross-validation) in Table 5.1. In addition, the performance of each baseline using the SURF with the voting approach (from Table 3.6) are included in the table for comparison. For each baseline, the best training and test results across all values

of k from Table 3.6 was chosen for both the Jaffe and UIUC datasets. This ensures the best SURF results are used for comparison.

As these results show, the histograms produced by GP are able to be used very effectively by other classifiers. Nearly all the classifiers achieve over 90% performance on the test set across the three different experiments. The SVM and Naïve Bayes perform comparably to the original GP tree, with accuracy within 2% on the test set. The performance is also comparable to the baselines which used SURF with voting. The SVM, Naïve Bayes and AdaBoost M1 classifiers all perform better using the features extracted by the GP tree than they did with the SURF features. When using SURF, the classifiers had a much larger number of features for classification ($k \times 64$), so the ability to achieve comparable performance demonstrates how GP-HoG can produce very good features.

Table 5.1: Performance of the extracted HoG histograms when used as features in other classifiers. Each classifier is given h features, where h is $8\times$ the number of histograms produced in the GP tree (i.e. number of occurences of the HoG function).

	Prog. 3 Features (Figure 4.1): Jaffe, $h = 16$.		Prog. 4 Features (Figure 4.2): Jaffe, $h = 16$.		Prog. 5 Features (Figure 4.3): UIUC, $h = 64$.		Jaffe: SURF + voting		UIUC: SURF + voting	
	Training	Test	Training	Test	Training	Test	Training	Test	Training	Test
GP tree	0.97	0.97	0.97	0.97	0.96	0.96	N/A	N/A	N/A	N/A
SVM	0.99	0.95	0.97	0.97	0.98	0.97	0.94	0.90	0.93	0.92
J48	0.98	0.89	0.99	0.93	0.99	0.94	1.00	0.97	1.00	0.94
NB	0.98	0.97	0.96	0.95	0.97	0.97	0.91	0.87	0.89	0.89
RF	1.00	0.95	1.00	0.93	1.00	0.98	1.00	0.98	1.00	0.96
ABM1	1.00	0.94	1.00	0.93	0.95	0.93	0.96	0.87	0.88	0.87

5.3 GP-HoG for Feature Construction

The previous section considered using GP for feature extraction by using the histograms produced by the HoG function in the GP tree as feature vectors. This idea can be extended further by using the values produced by the histogramVal or compHistogram functions, i.e. individual bin values of histograms or differences between histograms, as selected by the GP tree. Using these values as features is using GP for feature construction, as new features are created which are more refined than using the whole histogram. Again, the values produced by the histogramVal and compHistogram functions are concatenated to give a feature vector. The length of this vector is h, where h is the sum of the number of histogramVal and compHistogram functions in the program. For example, Figure 4.1 contains two histogramVal functions which return the values at $histogramA[6]$ and $histogramB[0]$. These two values would be concatenated to give a feature vector of the form $\left[histogramA[6], histogramB[0]\right]$.

This approach was applied to the same example programs using the same n-fold cross-validation approach as in Section 5.2. The results are shown in Table 5.2. As before, the baseline results using SURF with voting are also included for comparison.

While performance is generally slightly worse than in Table 5.1, all classifiers achieve over 90% test performance across all of the runs. Again the SVM, Naïve Bayes and AdaBoost M1 classifiers all perform better using the features constructed by GP-HoG than those produced by SURF. This is an impressive result, as only a very small number of features ($2 - 5$) are used for classification. This shows that the GP-HoG approach can produce very useful features which perform consistently across the whole dataset. Providing fewer features to these classifiers may also improve performance, as it lowers the chance of over-training

while also reducing the complexity of the search space the classifiers must train in. Having a large number of features (as in SURF without voting) can negatively affect performance as classifiers are unable to efficiently find good solutions, so using GP-HoG for feature construction to give more advanced features allows for simpler classifiers which are still very accurate.

Table 5.2: Performance of the constructed HoG histogram values when used as features in other classifiers. Each classifier is given h features, where h is the number of features constructed by the GP tree (i.e. sum of number of occurrences of the histogramVal and compHistogram functions).

	Prog. 3 Features (Figure 4.1): Jaffe, $h = 2$.		Prog. 4 Features (Figure 4.2): Jaffe, $h = 2$.		Prog. 5 Features (Figure 4.3): UIUC, $h = 5$.		Jaffe: SURF + voting		UIUC: SURF + voting	
	Training	Test	Training	Test	Training	Test	Training	Test	Training	Test
GP tree	0.97	0.97	0.97	0.97	0.96	0.96	N/A	N/A	N/A	N/A
SVM	0.92	0.94	0.98	0.98	0.97	0.97	0.94	0.90	0.93	0.92
J48	0.98	0.92	0.99	0.93	0.98	0.93	1.00	0.97	1.00	0.94
NB	0.95	0.94	0.97	0.97	0.96	0.96	0.91	0.87	0.89	0.89
RF	1.00	0.93	1.00	0.90	1.00	0.96	1.00	0.98	1.00	0.96
ABM1	1.00	0.92	1.00	0.90	0.96	0.94	0.96	0.87	0.88	0.87

5.4 Chapter Summary

This chapter showed that the GP-HoG approach was able to automatically extract and construct useful, high-level features which could be used by other classifiers for highly-accurate image classification. On the Jaffe dataset, classifiers could easily achieve over 90% test performance by using only two constructed features. This level of performance was similar to that achieved when SURF keypoints were used, despite the SURF + voting technique using a much larger number of features. This shows that the GP-HoG approach can be used with existing classifiers to enable very efficient image classification.

Chapter 6

Conclusions and Future Work

The work presented in this report aimed to develop a new domain-independent image classification by combining GP with existing feature extraction algorithms. A number of approaches were proposed which give promising results when compared to existing machine learning baselines, hence achieving this goal. Two main approaches used SURF and HoG respectively with GP techniques in different ways. The Hybrid GP-SURF approach considered a two-stage approach where GP was used to enhance the SURF algorithm, whereas the GP-HoG approach directly utilised an adaptation of the HoG algorithm within GP. These two different approaches present two novel ways of using GP in conjunction with high-level feature extraction for image classification, an area of research that has not been explored in depth. Both approaches were shown to be effective, but the second approach has the largest amount of potential as it allows the evolutionary process to train all of the region selection, feature extraction, and classification processes simultaneously.

6.1 Major Conclusions

A number of major conclusions can be drawn as a result of this work:

1. A hybrid method can be designed that combines GP and SURF to give promising performance results when compared to a range of well-known machine learning methods. Using GP to select regions for feature extraction and an SVM with a voting system to classify these features gave high classification accuracy across different datasets. This showed that GP is able to be used to produce more refined high-level features than those produced by the standard SURF algorithm. It was also shown that GP can have some success when used to classify SURF keypoint descriptors if a voting approach is used, but that other classifiers still give better performance when they use a voting approach. The improvement gained by the baseline classifiers when using voting was also an interesting finding presented in this report. It also was found that an overly complex function and terminal set would reduce GP performance rather than improving it, due to increasing the search space.

2. It was found that using GP for region selection, feature extraction, and classification within a single program could give good results. In particular, using more advanced feature extraction functions gave high classification accuracy than in the 2TGP approach which used simpler statistical features. The GP-HoG technique was also able to perform comparably when compared to the baseline methods which used a larger number of features, and has the potential to be further improved with the addition of functions inspired by other high-level feature extractors.

3. It was shown that the features extracted and constructed by good programs using the GP-HoG approach could be used as inputs to other classifiers with a high rate of accuracy. This showed how GP can be used very effectively for feature extraction and construction, as the GP-HoG approach produced features which could achieve over 90% accuracy using only two to five features on a range of existing classifiers.

4. The GP trees produced in this work were shown to be relatively easily understood by humans, giving an insight into how region selection can be used to improve classification performance. This showed that the methods proposed in this work were effective, as they produced solutions which could be analysed and understood.

6.2 Future Work

There are several extensions to this work that could be investigated. The Hybrid GP-SURF approach which used GP for region selection also used an SVM for performing classification of keypoints as part of a voting scheme. The results in Table 3.6 showed that the J48 and Random Forest classifiers responded the best to the voting scheme, and so may give better performance than an SVM as part of the GP-SURF approach. Investigating using different classifiers could improve results.

The GP-HoG approach used the HoG algorithm as inspiration for GP functions that perform high-level feature extraction. There are a number of other algorithms which could also be adapted to be used 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.

Using GP for for feature extraction and feature construction can also be further investigated. The analysis in Chapter 5 showed that the GP-HoG approach could perform very well in this regard. Using a wrapper approach where GP-HoG is used for feature construction and another classifier is used for classification would be an interesting direction to pursue.

The approaches in this work considered only binary classification tasks. Many image analysis problems have many classes (e.g. digit recognition). These techniques could be extended to multi-class classification by using techniques such as k-Nearest Neighbour (k-NN), or by using multiple thresholds corresponding to each class. Furthermore, this technique has only been applied to images containing a single, centred object; it would be more useful if these ideas could be extended to work on instances where there can be multiple objects in different locations. Such an approach could use a sliding window, where regions of the window are extracted instead of regions of the whole image.

Bibliography

[1] AGARWAL, S., AWAN, A., AND ROTH, D. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 11 (2004), 1475–1490.

[2] AL-SAHAF, H., SONG, A., NESHATIAN, K., AND ZHANG, M. Two-tier genetic programming: Towards raw pixel-based image classification. *Expert Systems with Applications 39*, 16 (2012), 12291–12301.

[3] AL-SAHAF, H., ZHANG, M., AND 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* (2014), ACM, pp. 84–89.

[4] AL-SAHAF, H., ZHANG, M., AND JOHNSTON, M. Binary image classification: A genetic programming approach to the problem of limited training instances. *Evolutionary Computation* (2015), 1–40.

[5] ALPAYDIN, E. *Introduction to machine learning*. MIT press, 2014.

[6] BACK, T., FOGEL, D. B., AND MICHALEWICZ, Z. *Handbook of evolutionary computation*. IOP Publishing Ltd., 1997.

[7] BALA, J., DE JONG, K., HUANG, J., VAFAIE, H., AND WECHSLER, H. Using learning to facilitate the evolution of features for recognizing visual concepts. *Evolutionary Computation 4*, 3 (1996), 297–311.

[8] BAY, H., TUYTELAARS, T., AND VAN GOOL, L. SURF: Speeded up robust features. In *Proceedings of the 9th European Conference on Computer Vision*. Springer, 2006, pp. 404–417.

[9] BHOWAN, U., ZHANG, M., AND JOHNSTON, M. Genetic programming for classification with unbalanced data. In *Proceedings of the 13th European Conference on Genetic Programming*. Springer, 2010, pp. 1–13.

[10] BRADLEY, A. P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition 30*, 7 (1997), 1145–1159.

[11] CSURKA, G., DANCE, C., FAN, L., WILLAMOWSKI, J., AND BRAY, C. Visual categorization with bags of keypoints. In *Proceedings of the Workshop on Statistical Learning in Computer Vision* (2004), vol. 1, Springer, pp. 1–16.

[12] DALAL, N., AND TRIGGS, B. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2005), vol. 1, IEEE, pp. 886–893.

[13] DÉNIZ, O., BUENO, G., SALIDO, J., AND DE LA TORRE, F. Face recognition using histograms of oriented gradients. *Pattern Recognition Letters 32*, 12 (2011), 1598–1603.

[14] EIBEN, A. E., AND SMITH, J. E. *Introduction to evolutionary computing*. Springer Science & Business Media, 2003.

[15] ESPEJO, P. G., VENTURA, S., AND HERRERA, F. A survey on the application of genetic programming to classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 40*, 2 (2010), 121–144.

[16] FARQUHAR, J., SZEDMAK, S., MENG, H., AND SHAWE-TAYLOR, J. Improving "bag-of-keypoints" image categorisation: Generative models and PDF-kernels. Tech. rep., University of Southampton, 2005.

[17] FAYYAD, U., PIATETSKY-SHAPIRO, G., AND SMYTH, P. From data mining to knowledge discovery in databases. *AI Magazine 17*, 3 (1996), 37–54.

[18] GHAMISI, P., AND BENEDIKTSSON, J. Feature selection based on hybridization of genetic algorithm and particle swarm optimization. *Geoscience and Remote Sensing Letters 12*, 2 (2015), 309–313.

[19] HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. H. The WEKA data mining software: an update. *SIGKDD explorations newsletter 11*, 1 (2009), 10–18.

[20] HARDING, S. Evolution of image filters on graphics processor units using cartesian genetic programming. In *Proceedings of the 10th IEEE Congress on Evolutionary Computation* (2008), IEEE, pp. 1921–1928.

[21] HINDMARSH, S., ANDREAE, P., AND ZHANG, M. Genetic programming for improving image descriptors generated using the scale-invariant feature transform. In *Proceedings of the 27th Conference on Image and Vision Computing New Zealand* (2012), ACM, pp. 85–90.

[22] HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[23] HUANG, Y., WU, Z., WANG, L., AND TAN, T. Feature coding in image classification: A comprehensive study. *IEEE Transactions on Pattern Analysis and Machine Intelligence 36*, 3 (2014), 493–506.

[24] KENNEDY, J., AND EBERHART, R. C. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks* (1995), vol. 4, IEEE, pp. 1942–1948.

[25] KENNEDY, J., EBERHART, R. C., AND SHI, Y. *Swarm intelligence*. Morgan Kaufmann, 2001.

[26] KHAN, N. Y., MCCANE, B., AND WYVILL, G. Sift and surf performance evaluation against various image deformations on benchmark dataset. In *Proceedings of the International Conference on Digital Image Computing Techniques and Applications* (2011), IEEE, pp. 501–506.

[27] KOLLER, D., AND SAHAMI, M. Toward optimal feature selection. In *Proceedings of the 13th International Conference on Machine Learning* (1995), pp. 284–292.

[28] KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.

[29] KRAWIEC, K. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines 3*, 4 (2002), 329–343.

[30] LI, S., WU, H., WAN, D., AND ZHU, J. An effective feature selection method for hyperspectral image classification based on genetic algorithm and support vector machine. *Knowledge-Based Systems 24*, 1 (2011), 40–48.

[31] LOWE, D. G. Object recognition from local scale-invariant features. In *Proceedings of the 7th IEEE International Conference on Computer Vision* (1999), vol. 2, IEEE, pp. 1150–1157.

[32] LUKE, S. *Essentials of Metaheuristics*, 2nd ed. Lulu, 2013.

[33] LYONS, M., AKAMATSU, S., KAMACHI, M., AND GYOBA, J. Coding facial expressions with gabor wavelets. In *Proceedings of the 3rd. International Conference on Face & Gesture Recognition* (1998), IEEE Computer Society, pp. 200–205.

[34] MAGHOUMI, M., AND ROSS, B. J. 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* (2014), IEEE, pp. 1–8.

[35] MIKOLAJCZYK, K., AND SCHMID, C. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 10 (2005), 1615–1630.

[36] MOLINARO, A. M., SIMON, R., AND PFEIFFER, R. M. Prediction error estimation: a comparison of resampling methods. *Bioinformatics 21*, 15 (2005), 3301–3307.

[37] MONTANA, D. J. Strongly typed genetic programming. *Evolutionary Computation 3*, 2 (1995), 199–230.

[38] MORENO, P., MARÍN-JIMÉNEZ, M. J., BERNARDINO, A., SANTOS-VICTOR, J., AND DE LA BLANCA, N. P. A comparative study of local descriptors for object category recognition: SIFT vs HMAX. In *Pattern Recognition and Image Analysis*. Springer, 2007, pp. 515–522.

[39] NENE, S. A., NAYAR, S. K., AND MURASE, H. Columbia object image library (COIL-20). Tech. Rep. CUCS-006-96, Columbia University, New York, 1996.

[40] NESHATIAN, K., ZHANG, M., AND ANDREAE, P. A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming. *IEEE Transactions on Evolutionary Computation 16*, 5 (2012), 645–661.

[41] OMRAN, M., SALMAN, A., AND ENGELBRECHT, A. P. Image classification using particle swarm optimization. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning* (2002), vol. 1, Springer, pp. 18–22.

[42] POLI, R. Genetic programming for image analysis. In *Proceedings of the 1st Annual Conference on Genetic Programming* (1996), MIT Press, pp. 363–368.

[43] POLI, R., LANGDON, W. B., MCPHEE, N. F., AND KOZA, J. R. *A field guide to genetic programming*. Lulu.com, 2008.

[44] RUSSELL, S., AND NORVIG, P. Artificial intelligence: a modern approach.

[45] SAINI, R., AND DUTTA, M. Image segmentation for uneven lighting images using adaptive thresholding and dynamic window based on incremental window growing approach. *International Journal of Computer Applications 56*, 13 (2012), 31–36.

[46] SERRE, T., WOLF, L., AND POGGIO, T. Object recognition with features inspired by visual cortex. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2005), vol. 2, IEEE, pp. 994–1000.

[47] SHAO, L., LIU, L., AND LI, X. Feature learning for image classification via multiobjective genetic programming. *IEEE Transactions on Neural Networks and Learning Systems 25*, 7 (2014), 1359–1371.

[48] WINKELER, J. F., AND MANJUNATH, B. Genetic programming for object detection. In *Proceedings of the 2nd Annual Conference on Genetic Programming* (1997), Morgan Kaufmann, pp. 330–335.

[49] WITTEN, I. H., AND FRANK, E. *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005.

[50] ZHANG, M., CIESIELSKI, V., AND ANDREAE, P. A domain-independent window approach to multiclass object detection using genetic programming. *EURASIP Journal on Advances in Signal Processing*, 8 (2003), 841–859.