Application of Multiobjective Genetic Programming to the Design of Robot Failure Recognition Systems

Yang Zhang and Peter I. Rockett

***Abstract*— We present an evolutionary approach using multiobjective genetic programming to derive optimal feature extraction pre-processing stages for robot failure detection. This data-driven machine learning method is compared both with conventional (non-evolutionary) classifiers and a set of domain-dependent feature extraction methods. We conclude MOGP is an effective and practical design method for failure recognition systems with enhanced recognition accuracy over conventional classifiers, independent of domain knowledge.**

***Note to Practitioners* — Detecting failures in robotic systems is critical to their autonomous operation since it allows mitigating/compensating control strategies to be brought into play. In this paper we employ multiobjective genetic programming to derive (near-)optimal feature extraction stages for failure detection which we demonstrate to yield significantly better detection performance than a range of comparator methods on benchmark datasets. Our approach requires no domain knowledge.**

***Index Terms*— Feature extraction, Autonomous robots, Failure recognition, Multiobjective genetic programming.**

I. Introduction

SINCE it is one of the key problems in autonomous robots, the design of failure recognition systems has been studied previously by a number of authors, for example [1]–[3].

Machine learning methods are invaluable for acquiring the complex knowledge associated with robotic tasks, especially fault detection and diagnosis [4]. Lopes & Camarinha-Matos [4] demonstrated that suitable transformations of the sensor measurements can improve prediction accuracy – a result which is well-established in the machine learning literature – although they used hand-crafted transformations based on domain knowledge. Their attempts at using the SMART+ algorithm [5] and oblique decision trees to construct new, linear features were generally unsuccessful.

As far as we are aware, all previous work on failure recognition has employed domain-specific knowledge which is probably sub-optimal and susceptible to inadvertent omission [4]. The principal contribution of the present paper is the application to robot failure detection of a generic, domain-independent machine learning methodology for automatically constructing (near-)optimal features to maximally separate the failure classes. Our main focus is on constructing new, more discriminatory features to be used within a classifier, hence our emphasis on *feature extraction* but it would be equally valid to regard this work as evolving (near-)optimal *classifiers*. We treat the recognition of failure in a robot as a multi-class classification problem.

As in [4] (and many others), the raw input to our classifier is a series of time-interval sampled sensor readings arranged into vectors. We confine ourselves to the generic task of *failure detection*, not the subsequent use of that information to failure recovery. Nonetheless, a number of critical issues need to be addressed:

- Firstly, a large number of robot sensors may need to be sampled over some time interval. The dimensionality of the raw input vector can thus be very high and selecting the optimal subset of features is a key sub-problem, both to improve recognition rates and to minimize detection latencies.
- Second, only limited ‘experience’ is available to the inductive learner which may leave the system prone to over-fitting, especially where sparse, high-dimensional data are concerned.
- Third, to meet the need to work in ‘real time’, we wish to design the most compact recognition system possible, subject to acceptable accuracy.

Like [4], we utilize feature extraction to improve classification accuracy but rather than using pre-defined – and probably suboptimal – transformations divined from domain knowledge, we propose explicitly *optimizing* the pre-processing stages to give the greatest attainable classification accuracy. We apply a generic feature extraction method [6], [7] which uses multiobjective genetic programming.

Genetic programming (GP) as an evolutionary computation technique has been applied to many fields including classifier design [8] and robot behavior design [9]. GP is an evolutionary technique in which each potential solution (or *chromosome*) in a population is generally represented as an acyclic directed graph. These graphs are interpreted as parse trees to yield a sequence of operations (or a program); internal nodes in the trees are mathematical functions while the terminal (leaf) nodes represent the input values of the problem. As in all evolutionary methods, members of the population are stochastically selected for breeding, biased in a measure of their performance on the problem; offspring are generated which can improve on the performance of the parents. GP has been described in a number of books; see [10], for example.

In this paper we present what we believe to be the first report of applying multiobjective genetic programming (MOGP) to the design of a robot failure recognition system. We evolve

Manuscript received June 9, 2008.

Yang Zhang is with the Laboratory for Image and Vision Engineering, Department of Electronic and Electrical Engineering, The University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK. {email: hegallis@gmail.com}

Peter Rockett is with the Laboratory for Image and Vision Engineering, Department of Electronic and Electrical Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD, UK. {email: p.rockett@shef.ac.uk}

domain-independent (near-)optimal feature extraction stages where the 'optimal' feature extractors learned during the training phase map the input patterns to a new, one-dimensional decision space in which the discriminability among classes is explicitly optimized. Classes can be optimally separated by a simple thresholding step in this 1D space. Since we are dealing here with multiple failure types, this basic method has been extended and details are presented in Section II.

This paper is organized as follows: We describe the evolution of feature extractors using MOGP in Section II together with the multi-class extensions used in this work. Experiments on the benchmark "Robot Execution Failures" datasets [11] are reported in Section III along with comparisons with previous studies on the same problem. Discussion of the extracted features and the optimized MOGP trees is also given. Conclusions to this work are presented in Section IV.

II. MOGP Feature Extraction

Evolutionary algorithms are not guaranteed to find a true mathematical optimum. We use the term "optimal" without the "(near-)" qualifier in the sense that it is widely understood in the evolutionary computing literature, to denote *approximately* optimal solutions. Notwithstanding this, evolutionary algorithms can produce demonstrably better solutions than other available methods.

The evolutionary strategy used here is an adaptation to genetic programming of the steady-state Pareto Converging Genetic Algorithm [12] and where the population comprises GP trees. We have used this evolutionary paradigm since it appears to give superior solutions to current competitor approaches [13].

A 2-class MOGP classifier can be derived by evolving a GP tree which maps the input variables into a 1D decision space; individual patterns are assigned a label by thresholding in this 1D space. Further, the distance between the point in the 1D space to which a pattern maps, and the threshold is a measure of confidence in the label assignment.

Our objective here is to identify a series of transformations that map the collected robot sensor data into a set of new, one-dimensional decision spaces in which the separation between the different failure types (*i.e.* classes) is maximized. The previous multiobjective MOGP approach [6], [7], [14] implements a 2-class classifier and our approach for dealing with the present q-class problem has been to decompose this into a series of 2-class problems.

Bailey [15] has discussed various strategies for decomposing a q-class problem into a series of 2-class problems. For MOGP, we have concluded that hierarchical decomposition gives the best results [16]. Given q failure types, we select one type, $\omega_i; i \in [1 \cdots q]$ and evolve a classifier to separate this class from the remaining $(q-1)$ classes. We then select one of the remaining $(q-1)$ classes and evolve a further set of feature extractors to separate that class from the other $(q-2)$ types. And so on. See [16] for further details.

Hierarchical decomposition requires q, 2-class classifiers. For example, to separate three classes: A, B and C, we first classify A versus $(B \wedge C)$. Next we separate B from C. Finally, we classify C from all the other classes. This final q-th step is required because, if in separating B and C, we regard everything which is not a B as a C we will produce a large number of errors – any instances of A misclassified by the first A versus $(B \wedge C)$ stage will be incorrectly assigned to class C by default.

One of the noted difficulties with decomposing multi-class problems into a series of 2-class problems is that some patterns are assigned no label at all. Error correcting codes have been used to resolve labeling ambiguities although here, we have used the normalized distances between the projection of each pattern into the 1D decision spaces, and the decision thresholds. Since this distance is a measure of confidence in a label, we have given unlabeled patterns the class corresponding to the closest approach to the relevant decision threshold – that is, the class to which they missed being assigned by the smallest margin. Subject to the assumption of mono-modal class-conditioned densities in the decision spaces, this approach can be interpreted as maximizing the posterior probability [16].

Importantly, no assumptions have been made about the statistical distributions of the original sensor data.

In terms of optimizing the pattern separability in the 1D decision spaces, we 'minimize' (see below) a three-dimensional vector of fitness objectives comprising:

1) Tree complexity measurement: The node count of the tree is used as the measure of solution complexity. Unless inhibited from doing so, trees in GP have a tendency to grow without limit, a phenomenon termed *bloat*. Bloat is undesirable for a number of reasons: Firstly, overly complex trees tend to generalize poorly, a manifestation of the well-known *overfitting* effect found in machine learning. Minimizing tree complexity accords with Occam's Razor. The second objection to bloated trees is that they require large evaluation times which slows the evolution (and any subsequent use online). It has been shown [17] that minimizing the tree node count in a multiobjective setting can effectively suppress tree bloat.

2) Misclassification error: For a given training set/step in the hierarchical decomposition, all the (n-dimensional) input patterns are projected into a 1D decision space in which there will inevitably be some class overlap. Within the evolutionary loop we use Golden Section search [18] in the 1D decision space to locate the optimum decision threshold by minimizing the number of training patterns misclassified by that particular feature transformation. Golden Section search is terminated when there is no further reduction in the error. The resulting *misclassification error* (fraction of misclassified training patterns) is the second of our three fitness objectives.

3) Bayes Error: Appropriate fitness functions are critical to the success of evolutionary algorithms and this has motivated our use of the third objective – an estimate of the Bayes error. Our feature mapping projects all patterns of the training set into the 1D decision space where they will form two, class-conditioned PDFs. We estimate the Bayes error – a fundamental lower bound on classification performance – by calculating the overlap between the histograms of the two class-conditioned PDFs; clearly we aim to minimize this quantity since it is a measure of class separability.

The reason for the apparent duplication of the misclassification error (see 2, above) stems from the fact that using the misclassification error alone usually results in very slow (or no) convergence. The misclassification error objective alone seems unable to exert sufficient selective pressure at the start of the evolutionary optimization when almost all of the randomly-created individuals in the population have very high misclassification errors; selective pressure is thus rather weak and the optimization proceeds slowly. The Bayes error, on the other hand, appears able to differentiate between individuals of slightly greater promise in the initial phases and its inclusion greatly speeds (or indeed facilitates) convergence [8]. (Conversely, using the Bayes error on its own can lead to poor generalization [7]. We have found that both misclassification error and the Bayes error are needed to produce good results.)

Each individual is evaluated against these three fitness objectives, each of which we wish to simultaneously minimize. In practice, minimizing all three is not possible since they are implicitly coupled – for example, individuals with large numbers of nodes tend to give lower (training set) misclassification error. In the past, such multiple objective optimization has been carried out by linearly weighting the three objectives but this inevitably biases the solution – there is no principled way of aggregating non-commensurable objectives. We thus 'minimize' the 3-vector of objectives using *Pareto optimality* which invokes a notion of vector dominance [19]. Given two N-dimensional vectors of objectives, $\mathbf{x}$ and $\mathbf{y}$, within a minimization problem, $\mathbf{x}$ is said to *dominate* $\mathbf{y}$ that is, $\mathbf{x} \prec \mathbf{y}$:

$$\mathbf{x} \prec \mathbf{y} \text{ iff } \forall x_i \leqslant y_i, i \in [1 \ldots N] \ \wedge \ \exists x_j < y_j, j \in [1 \ldots N]$$

This Pareto dominance relation can be used for comparing and ranking solutions, and hence is the basis for selection for breeding on the grounds of fitness within the MOGP.

What results from this MOGP optimization is a *set* of solutions, each member of which is equivalent in the sense that although each of the elements of the objective vector have different values, no solution in this set can be considered superior to any other. The members of this so-called *Pareto set* all represent 'optimal' solutions since for any member, it is not possible to reduce the value of any one objective without simultaneously increasing the value of at least one of the other objectives. In the field of pattern recognition, we are principally concerned with obtaining the lowest possible misclassification error and so here, we adopt the solution which has the best score on this attribute. Nonetheless, the other two objectives play a critical role in guiding the search for the ultimately selected solution.

The MOGP parameters used in this work are summarized in Table I. The initial population was randomly created with half the trees of full depth (here, 5) and half of random depth in the range $[1 \ldots 5]$ although thereafter, there was no explicit limitation on the depth to which trees could grow other than that implicitly imposed by the tree complexity objective.

III. Experiments and Results

To verify our proposed method, we have applied it to the real-world robot failure recognition datasets described in [4], [11]. The data comprise three force sensor measurements

TABLE I
GP Settings

Terminal set	90 dimensional sensor vectors
	Floating point numbers $\in [0.0 \ldots 1.0]$
Function set	sqrt, log, pow2, - (unary minus), sin
	-, +, $\times$, $\div$, max, min, xor
	if-then-else
Population size	250
Original population	Half full trees, half random trees
Original max. tree depth	5
Stopping criterion	10,000 function evaluations

(f_x, f_y, f_z) and three torque sensor measurements (t_x, t_y, t_z) each sampled at fifteen regularly-spaced time intervals, concatenated into ($6 \times 15 =$) 90 dimensional pattern vectors. Each pattern vector thus comprises a labeled instance of a multivariate time profile of force and torque values. The recognition task is to identify the particular failure associated with each sensor measurement profile.

There are five separate datasets:

- LP1: 88 instances from 3 failure types and normal type (failures in approach to the grasp position)
- LP2: 47 instances from 4 failure types and normal type (failures in the transfer of a part)
- LP3: 47 instances represent 3 failure types and normal type (position of part after a transfer failure)
- LP4: 117 failures from 2 failure types and normal type (failures in approach to the ungrasp position)
- LP5: 164 instances from 4 failure types and normal type (failures in motion with a part)

We applied seven conventional classifiers to the datasets as well as the MOGP-generated classifier. The implementations of the seven comparator algorithms employed were all taken from the Weka machine learning system[1] [20] and the default parameter settings were used except where noted below. The classifiers used were:

- RBF: Radial Basis Functions, a normalized Gaussian radial basis function network using the k-means clustering algorithm. We estimated the 'optimal' number of clusters (k) for each problem by randomly splitting each dataset, using one partition as a training set and the other as a validation set. For that dataset we adopted the value of k which gave the lowest validation error.
- MC: MultiClass Classifier, a multinomial logistic regression model using a ridge estimator as the base classifier. Handles multi-class problems with 2-class classifiers and error correcting output codes for increased accuracy.
- NNge: nearest neighbor algorithm using non-nested generalized exemplars.
- BN: BayesNet, Bayes network classifier using the K2 learning algorithm.
- IB1: instance-based learning algorithm. The distance measures are used to find the training instance closest to the test instance and predict the same class as the training instance.

[1]See: `http://www.cs.waikato.ac.nz/~ml/weka/`. We have used Version 3.4.5 of Weka in this work.

- SMO: sequential minimal optimization algorithm to train a support vector classifier.
- C4.5: the well-known decision tree algorithm (referred to as “J48” in Weka.)

Conventionally classifiers have been compared with N-fold cross-validation on a dataset followed by a t-test to determine the statistical significance of the differences in the observed error rates. This, however, is unsound [21], Consequently we use an empirical 5×2 *cv* F-test due to Alpaydin [21], [22] at the 95% confidence level.

The mean percentage errors averaged over 10 folds for each dataset are shown in Table II for every classifier considered here (the seven conventional classifiers and the MOGP classifier) using the 90D raw pattern data. It is noteworthy that the MOGP-derived classifier returns the smallest error value on every dataset (LP1-5) although the statistical significance of this needs to be established separately using the 5×2 *cv* F-test. The set of pairwise statistical comparisons between the MOGP classifier and each conventional comparator is summarized in Table III; a tick indicates that MOGP is superior whereas a dash indicates that there is no discernible difference. From Table III, MOGP is statistically superior on 34 of the 35 comparisons and yields statistically identical results only on the RBF/LP1 pairing; notably the RBF classifier yields variable results on the other datasets – see Table II. The benefits of the MOGP approach are therefore obvious.

The evolved feature extracting trees are typically rather simple transformations – the largest tree is shown in Fig. 1 – and typically of depth around four. MOGP is thus able to produce feature transformations which directly address requirement for low latency classification. Similarly, due to the parsimony pressures in the algorithm, the MOGP trees typically use a modestly-sized subset of the 90 possible inputs, in the range $[1 \ldots 7]$ raw attributes, again directly addressing the latency requirements.

TABLE III
F-TEST COMPARISON BETWEEN MOGP AND CONVENTIONAL ALGORITHMS FOR THE FIVE DATASETS. SEE TEXT FOR DETAILS.

Datasets	MOGP						
	RBF	MC	NNge	BN	IB1	SMO	C4.5
LP1	-	✓	✓	✓	✓	✓	✓
LP2	✓	✓	✓	✓	✓	✓	✓
LP3	✓	✓	✓	✓	✓	✓	✓
LP4	✓	✓	✓	✓	✓	✓	✓
LP5	✓	✓	✓	✓	✓	✓	✓

The datasets used here were previously studied by Lopes & Camarinha-Matos [4] who investigated the effects on classification accuracy of five hand-crafted feature extraction methods, referred to as S1 to S5. S1 uses the measured force and torque values as raw features without preprocessing giving 90D input vectors; S2 includes the amplitudes of forces and torques in the orthogonal planes in 3D space in addition to the raw measurements giving 210D input vectors; S3 extracts summary features into 72D vectors; S4 extracts the amplitudes of the Fourier transforms of the time series samples to yield 48D input vectors; S5 jointly uses all the features mentioned above concatenated into 490D vectors. Full details can be found in [4]. In Table IV we reproduce the classification results of [4] alongside the MOGP results although we stress these two sets of figures are not *directly* comparable. In [4] the authors assessed performance using a leave-one-out method; that is, for a dataset with n data, n classifiers were trained on $(n-1)$ data and each tested on the omitted n-th datum. The error is returned as the average over n repetitions of training.

Here our MOGP validation errors have been estimated from the average over ten, 50-50 splits of the datasets, training on only $n/2$ data. Thus if all other things are equal, the errors obtained by Lopes & Camarinha-Matos should be significantly lower since their classifiers were trained on almost twice as many data ($n-1$ *vs.* $n/2$) although quantifying this advantage is difficult. (Repeating the leave-one-out methodology here for direct comparison with [4] is problematic since to separate q classes requires q, 2-class classifiers. Thus to obtain the leave-one-our error estimate for, say, LP1 would require 88 data $\times$ 4 classes $= 352$ classifier trainings. To process all the datasets, LP1-5 would require the training of around two thousand 2-class classifiers which is impractical.)

Although the error values in Table IV from MOGP and from [4] are not directly comparable due to the difference in methodologies, the results in [4] are trained on twice as many data and should therefore be more accurate. Apart from the LP1/S3 combination, the errors are not that much smaller and possibly not by any statistically significant amount. Indeed for the LP2 dataset, MOGP returns the lowest error by around a factor of three. The conclusion we draw is that despite being trained on half the data, MOGP is a more consistent performer, with error rates close to the best obtained in [4] for all datasets. Pre-processing strategy S3 gives the best overall results of the hand-crafted approaches but also returns the second worst error rate (49%) on LP2. MOGP, on the other hand is ranked either first, second or in one case, third for each dataset. Additionally, we reiterate that our MOGP approach requires no domain knowledge.

TABLE IV
MEAN PERCENTAGE ERRORS FOR LP1-5 FOR THE FEATURE EXTRACTION METHODS IN [4] AND MOGP. LOWEST ERRORS SHOWN IN BOLD TYPE.

	S1	S2	S3	S4	S5	MOGP
LP1	22	20	**4**	15	11	9.09
LP2	55	43	49	32	36	**15.30**
LP3	51	25	**13**	15	17	16.67
LP4	35	40	**5**	23	17	6.77
LP5	31	37	28	51	**23**	26.82

IV. CONCLUSIONS

In this paper we have hierarchically applied multiobjective genetic programming (MOGP) to optimize the feature extraction stages of a robot failure recognition system. Our domain-independent framework has demonstrated its superiority (or, at worst, statistical equivalence) to seven popular conventional classifiers over the five investigated datasets by displaying consistently smaller misclassification errors. Compared to the results from hand-crafted feature construction

TABLE II
5 × 2 CV MEAN PERCENTAGE ERROR COMPARISONS OF CLASSIFIERS ON THE FIVE DATASETS. (SMALLEST ERRORS SHOWN IN BOLD FACE.)

Dataset	Classifier							
	RBF	MC	NNge	BN	IB1	SMO	C4.5	MOGP
LP1	**9.09**	40.91	52.27	20.46	11.36	54.55	27.27	**9.09**
LP2	56.52	60.87	60.87	43.48	65.22	60.87	47.83	**15.30**
LP3	37.5	37.5	37.5	58.33	50.00	37.5	45.83	**16.67**
LP4	27.12	28.81	30.51	8.48	22.03	25.42	25.42	**6.77**
LP5	32.93	56.10	40.24	40.24	42.68	63.41	56.10	**26.82**

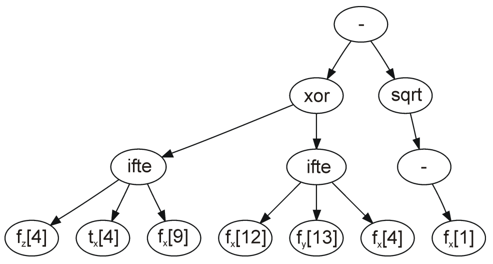

Fig. 1. Evolved GP tree for the ‘normal’ type for the LP2 dataset

stages of Lopes & Camarinha-Matos [4], our results are highly competitive although differences in methodology mean that direct comparison is not possible; the performance figures in [4] were obtained using almost twice as much training data although the expected improvement in misclassification error is not great. More particularly, our method uses no domain knowledge. MOGP feature extraction is thus confirmed as a highly promising approach to the design of failure detection, and potentially other, robotic sub-systems.

Since our MOGP framework requires no human intervention (other than providing a labeled training set), if additional data become available then the failure detection system can be ‘refined’ by continuous evolution, possibly executing as a background process. If superior feature extraction stages are identified they could be ‘switched’ into use, thus upgrading system performance. This extension, however, requires a careful study of incremental learning in multiobjective GP which will be the subject of future research.

ACKNOWLEDGMENT

Yang Zhang would like to thank for the financial support of a Universities UK Overseas Research Student Award Scheme (ORSAS) scholarship and the Henry Lester Trust.

REFERENCES

[1] C. Ferrrell, “Failure recognition and fault tolerance of an autonomous robot,” *Adaptive Behavior*, vol. 2, no. 4, pp. 375–398, 1994.

[2] B. Lussier, R. Chatila, F. Ingrand, M.-O. Killijan, and D. Powell, “On fault tolerance and robustness in autonomous systems,” in *3rd IARP-IEEE/RAS-EURON Joint Workshop on Technical Challenges for Dependable Robots in Human Environments*, Salford, UK, 7-9 September 2004.

[3] P. Goel, G. Dedeoglu, S. I. Roumeliotis, and G. S. Sukhatme, “Fault detection and identification in a mobile robot using multiple model estimation and neural network,” in *IEEE International Conference on Robotics and Automation (ICRA’00)*, San Francisco, CA, 22-28 April 2000, pp. 2302–2309.

[4] L. S. Lopes and L. M. Camarinha-Matos, “Feature transformation strategies for a robot learning problem,” in *Feature Extraction, Construction and Selection: A Data Mining Perspective*, H. Liu and H. Motoda, Eds. Boston: Kluwer Academic Publishers, 1998, pp. 375–391.

[5] M. Botta and A. Giordana, “SMART+: A multi-strategy learning tool,” in *13th International Joint Conference on Artificial Intelligence*, Chambrey, France, 28 August - 3 September 1993, pp. 937–943.

[6] Y. Zhang and P. I. Rockett, “Feature extraction using multi-objective genetic programming,” in *Multi-Objective Machine Learning*, Y. Jin, Ed. Heidelberg: Springer, 2006, pp. 75–99.

[7] ——, “A generic optimal feature extraction method using multiobjective genetic programming,” Submitted to *Applied Soft Computing*, 2007.

[8] T. Loveard and V. Ciesielski, “Representing classification problems in genetic programming,” in *Congress on Evolutionary Computation*, Gangnam-gu, Seoul, Korea, May 2001, pp. 1070–1077.

[9] K. J. Lee and B. T. Zhang, “Learning robot behaviors by evolving genetic programs,” in *26th International Conference on Industrial Electronics, Control and Instrumentation (IECON-2000)*, Nagoya, Japan, 22-28 October 2000, pp. 2867–2872.

[10] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming - An Introduction. On the Automatic Evolution of Computer Programs and Its Applications*. San Francisco, CA: Morgan Kaufmann, 1998.

[11] S. Hettich and S. D. Bay, “The UCI KDD archive,” Department of Information and Computer Science, University of California, Irvine, CA. http://kdd.ics.uci.edu, 1999.

[12] R. Kumar and P. I. Rockett, “Improved sampling of the Pareto-front in multiobjective genetic optimizations by steady-state evolution: A Pareto converging genetic algorithm,” *Evolutionary Computation*, vol. 10, no. 3, pp. 283–314, 2002.

[13] Y. Zhang and P. I. Rockett, “Comparison of evolutionary strategies for multi-objective genetic programming,” in *IEEE Systems, Man Cybernetics Society Conference on Advances in Cybernetic Systems (AICS2006)*, Sheffield, UK, 7-8 September 2006.

[14] ——, “Evolving optimal feature extraction using multi-objective genetic programming: A methodology and preliminary study on edge detection,” in *Genetic and Evolutionary Computation Conference (GECCO 2005)*, Washington, DC, 25-29 June 2005, pp. 795–802.

[15] A. Bailey, “Class-dependent features and multicategory classification,” Ph.D. dissertation, Department of Electronics and Computer Science, University of Southampton, Southampton, UK, 2001.

[16] Y. Zhang and P. I. Rockett, “Domain-independent approaches to optimize feature extraction for multi-classification using multi-objective genetic programming,” Submitted to *Pattern Analysis and Its Applications*, 2007.

[17] A. Ekárt and S. Z. Németh, “Selection based on the Pareto nondomination criterion for controlling code growth in genetic programming,” *Genetic Programming and Evolvable Machines*, vol. 2, no. 1, pp. 61–73, 2001.

[18] M. T. Heath, *Scientific Computing: An Introductory Survey*. New York: McGraw-Hill, 1997.

[19] C. A. C. Coello, “An updated survey of GA-based multiobjective optimization techniques,” *ACM Computing Surveys*, vol. 32, no. 2, pp. 109–143, 2000.

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

[21] T. Dietterich, “Approximate statistical tests for comparing supervised classification learning algorithms,” *Neural Computation*, vol. 10, no. 7, pp. 1895–1923, 1998.

[22] E. Alpaydin, “Combined 5 × 2 cv F-test for comparing supervised classification learning algorithms,” *Neural Computation*, vol. 11, no. 8, pp. 1885–1892, 1999.