Edge Detector Evolution using Multidimensional Multiobjective Genetic Programming

Y Zhang and P I Rockett

Technical Report No. VIE 2006/003
Department of Electronic and Electrical Engineering
University of Sheffield

Edge Detector Evolution using Multidimensional Multiobjective Genetic Programming

Yang Zhang hegallis@gmail.com
Peter I. Rockett p.rockett@shef.ac.uk
Laboratory for Image and Vision Engineering, Department of Electronic and Electrical Engineering, University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK

Abstract

In this paper we report the evolution of a feature extraction stage for edge detection using multidimensional multiobjective genetic programming. We have employed training and validation data produced using a realistic model of the imaging physics to evolve an n^2-to-m mapping which projects the pixel intensities of an $n \times n$ image patch into an m-dimensional decision space. The (near-)optimal value of m is also simultaneously determined during evolution. A conventional Fisher linear discriminant is then used to classify edge patterns. On the independent validation set, the suggested edge detector is shown to give performance superior to both the well-known conventional Canny detector and to earlier multiobjective genetic programming results which projected the pattern vector into a one-dimensional decision space. In addition, the superiority of the new detector is also demonstrated on a hand-labeled set of real images.

Keywords

Genetic programming, edge detection, multidimensional multiobjective optimization, edge labeling performance.

1 Introduction

Edge detection is a fundamental operation in image processing and computer vision which often serves as the first stage for more sophisticated algorithms, such as object recognition. Although 'global' Bayesian methods have been explored – for example, (Tsai et al., 2001) – these are complex and time-consuming therefore local methods which label an image using only the information contained in an $n \times n$ region of support remain the overwhelmingly preferred approach of image processing practitioners. (Typically n is odd and in the range 3-9.)

Edge detection has been an active topic of research for over 40 years and more recently has become intertwined with work on the performance assessment of computer vision algorithms. The common conclusion reached by these performance studies, e.g. (Bowyer et al., 2001; Heath et al., 1998), is that the edge detection algorithm due to Canny (1986) is the best currently available method. Canny's algorithm derives an optimized linear filtering kernel under the assumptions of a discontinuous step in image intensity and white noise. By maximizing a function made-up of a localization term and a signal-to-noise term, Canny proposed the derivative of a Gaussian as close to the optimal kernel and added two additional post-processing stages: non-maximal

suppression (NMS) and hysteresis thresholding; the now commonly adopted version of Canny's algorithm is summarized by Jain et al. (1995).

Although several performance evaluation studies (Bowyer et al., 2001; Heath et al., 1998) have confirmed the commonly held sentiment that the Canny algorithm is the best currently available edge detection algorithm, very recently Zhang and Rockett (2006e) have examined the Bayesian (i.e. minimum risk) operating point of the Canny detector. Zhang & Rockett concluded that the much-vaunted linear filtering stage – which is the central topic of Canny's 1986 paper – produces a very poor detector, yielding error rates that are barely lower than a priori declaring all pixels to be non-edges without considering the data at all. These authors concluded that the principal contributors to the performance of the Canny detector were the post-processing stages of NMS and hysteresis thresholding which appear to have been added by Canny almost as an afterthought. Viewed from a pattern recognition standpoint, there is thus much scope for improving on the state-of-the-art in local edge detection.

A number of workers have attempted to improve on Canny's detection algorithm using evolutionary methods. Jalali and Boyce (1995) employed a genetic algorithm to evolve an improved filtering kernel by optimizing Canny's localization/detection criterion. Harris and Buxton (1996) used genetic programming to devise a series of transformations to detect edges, again optimizing Canny's criterion although curiously, Harris and Buxton terminated evolution when they attained a performance equivalent to Canny's. More recently, Zhang and Rockett (2005) have used multiobjective genetic programming (MOGP) to derive an edge detector with demonstrably improved labeling performance over the Canny algorithm. This work was further refined in (Zhang and Rockett, 2006b) where it was concluded that the steady-state PCGA-based evolutionary algorithm of Kumar and Rockett (2002) was able to find more compact mappings than the SPEA2-based approach of Bleuler et al. (2001).

Framed as a problem in statistical pattern recognition, the classical edge detection approach comprises: A feature extraction stage which projects the $n \times n$ image patch into a one-dimensional space in which a simple threshold classifier determines whether or not to label the central pixel of the odd-sized region of support as an edgel (edge element). This is depicted in Figure 1 where, in common with the usual practice in pattern recognition, the feature extraction stages for edge detection have hitherto been deduced from domain-specific knowledge. For example, since edges are typically rapid spatial changes in image intensity, numerical approximations to spatial derivatives are frequently employed. Also, since the detection problem takes place in the presence of noise, various filters have been used although as Petrou and Bosdogianni (1999) have pointed-out, the challenge comes in removing noise without also removing excessive amounts of the underlying high-frequency image information. For the reasons discussed in Zhang and Rockett (2006e), convolutional approaches to edge detection appear limited which possibly explains why there has been no particularly significant performance advance in the last twenty years. Furthermore, Canny-like criteria are analytically tractable proxies for the ultimate desired goal of maximizing the labeling performance – which is a discrete problem and therefore not amenable to the methods of classical optimization.

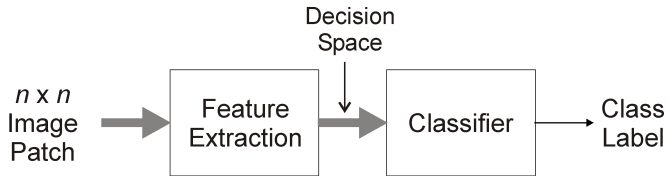

Figure 1: Prototypical edge detection system

In terms of applying statistical pattern recognition to edge detection, Zheng et al. (2004) have reported using a support vector machine (SVM) in which they embedded the first- and second-order derivatives of image intensity into the kernel mapping. It is noteworthy, however, that despite using a powerful classification paradigm in the SVM, these workers report a disappointing performance of only about the same as the Canny detector; we suspect that in implicitly using a linear filter, the performance of the SVM was limited by what appears to be the fundamental limit of convolutional methods (Zhang and Rockett, 2006e).

In this paper we report a significant extension to our previous work on deriving edge detectors using multiobjective genetic programming (Zhang and Rockett, 2005). In the previous work we evolved a feature extraction stage which projected the $n \times n$ image patch into a 1D decision space – see Figure 1. In some sense, the induced feature extraction stage in (Zhang and Rockett, 2005) was a 'drop-in' replacement for the "Feature extraction" stage in Figure 1. In the present work we project the $n \times n$ image patch into an m dimensional decision space where m – the 'optimal' decision space dimensionality, which is unknown – is determined as part of the evolutionary optimization. The fact that the edge labeling decision is being made in a more discriminatory, m-dimensional decision space instead of all the information being projected down into a 1D decision space has resulted in improved performance. We stress that we perform no other processing on the raw image pixel intensities – these unmodified quantities form the direct inputs to our feature extraction stage.

In Section 2 we describe the method for generating the training and test data used in this work and in Section 3 we describe the multidimensional multiobjective genetic programming method employed. We present a comparison of results in Section 4 in which we show that the present results are not only an improvement over Canny but also over the 1D decision space MOGP results in (Zhang and Rockett, 2005). We also present results on the hand-labeled USF image sets of Bowyer et al. (2001).

We offer conclusions to the present work in Section 5 as well as making some suggestions for future work.

Finally, mindful that image processing practitioners cannot reasonably repeat the computations described here in order to derive useful edge detectors, we present full implementation details of our suggested edge detector together with numerical values in an Appendix to this paper.

2 The Data Model

Clearly the performance of any inductive learning algorithm will reflect the quality of the data upon which it was trained. In the present context of training an edge detector, we have basically two choices for a training dataset: Hand labeling of real imagery or constructing a synthetic dataset based on physical optics.

In terms of the first option, Chen et al. (1996) examined hand-labeled datasets for training neural network edge detectors and concluded that such a dataset produced poor learning, principally because the examples did not adequately cover the pattern space. This can be understood from the fact that real imagery contains coherent identifiable objects, the edges from which are correlated, leading to a limited sampling of the pattern space. (These correlations between edge patterns are, of course, the basis for recognizing objects.) Further, hand labeling is a somewhat subjective process and therefore the resulting dataset is likely to contain significant numbers of errors which will impede learning; Zhang and Rockett (2006e) have shown that the hand-labeled USF dataset (Bowyer et al., 2001) contains numerous errors.

As to generating a synthetic dataset based on a step edge in image intensity, there is a sentiment in some sections of the image processing community which says that "real" edges are not step-like and that any results obtained using synthetic data "have limited value" (Zhou et al., 1989; Heath et al., 1998). We consider this blanket criticism of synthetic data to be misguided. Although step-edge models such as those used by Abdou and Pratt (1979) and Lyvers and Mitchell (1988) are naïve, they do represent an *approximation* to the discontinuities in image intensity which occur at an edge. (Ironically, critics also ignore the fact that Canny's algorithm is derived from a step edge in continuous space. We do not, however, deride this since it is an eminently sensible approximation for Canny to have used in his convolutional approach). Critics of synthetic datasets also opine that they fail to account for edges in textured regions: we know of no conventional edge detector which has assumed anything other than an edge comprising two semi-infinite half planes of uniform intensity. Thus to reject synthetic data for failing to model effects are not included in alternative analytic approaches lacks objectivity. That said, the abrupt step edge models used in (Abdou & Pratt, 1979; Lyvers & Mitchell, 1988) are overly simplistic and we employ here a much more sophisticated and complete model of the imaging physics to generate the datasets. Notwithstanding, Section 4 does include results on the hand-labeled USF datasets (Bowyer et al., 2001) for the sake of completeness.

We start from the assumption that an edge can be defined as an abrupt change in intensity in a gray-level image. Such a knife-edge model has long been used in optics. Our imaging model commences with the projection of an abrupt step edge onto the (presumed) focal plane of a CCD camera; the step is vertical and passes through the center of the central cell of an odd-sized region of support. See Fig 2(a). Since we have no prior reason for assuming that an edge has any particular orientation or alignment relative to the center of the region of support, we generate each instance of a dataset pattern by applying a randomly chosen rotation in the range $[0 \ldots 2\pi]$ followed by a random translation of the edge by $\Delta x, y \in [-1.5 \ldots +1.5]$ pixels, where the pixels in Figure 2 have unit dimension. Since we make no prior assumptions about the geometry of the edge, all of the random transformations described above are uniformly distributed. The image intensities on either side of the edge are

randomly chosen (as floating point quantities) from uniformly-distributed values in the range $[0 \ldots 255]$, again because we have no prior preference about image intensities.

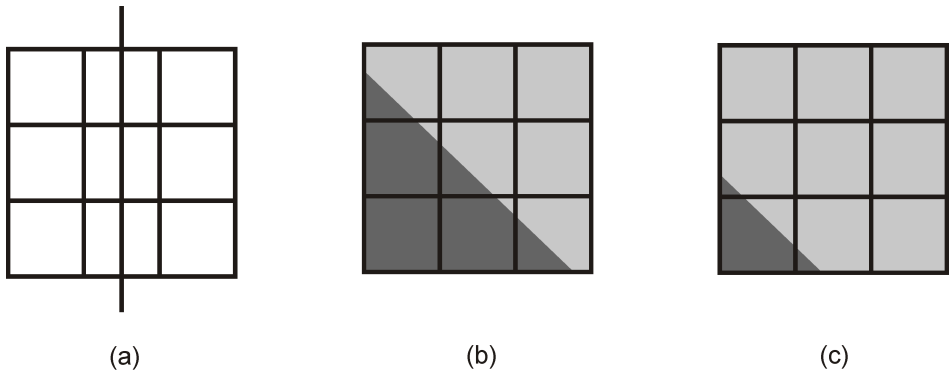

Figure 2: (a) Initial orientation of the knife edge; (b) A typical edge pattern; (c) A typical non-obvious non-edge pattern. Note that for illustration purposes only, the image patches are shown as 3×3; in actuality, patches of 13×13 were used.

At this point we assign a label to the pattern generated above: if, after affine rotation/translation, the edge still passes through the central cell of the region of support (Figure 2(b)) we label the pattern as an 'edge'; our objective is to label pixels which are intersected by an edge. Conversely, if after affine transformation, the edge does *not* pass through the central cell (Figure 2(c)) we label this pattern as a non-edge. (In keeping with the terminology of (Chen et al., 1996), we describe this latter situation as a "non-obvious non-edge" since although the pattern is clearly edge-like, we do not wish to label it as an edge because it does not pass through the central cell). It transpires that non-obvious non-edges (NONEs) are the principal source of error in edge detection (Chen et al., 1996; Zhang and Rockett, 2006e).

Up to this stage, the edge is (implicitly) assumed to have been imaged by a perfect optical system. Although the degradations introduced by diffraction-limited lenses are well-understood (Born and Wolf, 1999; Rockett, 2003), here we have convolved the image patch with a 2D Gaussian ($\sigma_{PSF} = 1$) to simulate effects of the point spread function (PSF) of the optical system. We have found this to be a good choice based on experimental observations.

In practice we have used a 13×13 region of support – almost certainly larger than is needed. Part of our motivation here is to explore whether the evolutionary procedure set-out in Section 3 selects a reasonable subset of pixels to construct an edge detector.

The next stage in generating a single edge/NONE pattern is to sample the transformed/blurred intensity function in Figure 2 over each pixel; in practice, this is carried-out by integration over the square pixel region to yield a 169-dimensional (13×13) pattern which correctly incorporates the effects of convolving the continuous image intensity function with the sampling function due to the pixel array. We assume here that the pixels in the CCD array exactly abut although this is not a fundamental

limitation of the model; indeed possible extension of this work to motion detection would probably need to include the effects of inter-pixel gaps. Finally, each of the (floating-point) pixel intensity values was quantized to one of 256 gray levels simulating the analogue-to-digital conversion process which takes place in framegrabbers.

In constructing the overall datasets, we repeatedly generate edge/NONE patterns with different, randomly-chosen orientations/translations/intensities which we supplement with image patches of uniform, randomly chosen gray-level. We term these supplementary uniform patches "obvious non-edges" since they are very obviously not edges.

Clearly to facilitate proper training, the composition of the training set has to reflect the prior probability of edges in real images and we have assumed the typical figure of 0.05 (5%) for the edge prior. From Figure 2 it is apparent that for every edge, there should be two non-obvious non-edges (Zhang and Rockett, 2005) and the training dataset was thus constructed with 10% NONEs. Consequently, each dataset consisted of 85% uniform patches (100% - 5% - 10% = 85%).

The training dataset comprised 10,000 patterns. The independently generated validation set consisted of 100,000 patterns. Unlike the training set, each of the pixel intensities in the validation set was corrupted with Gaussian-distributed noise ($\sigma_N = 2$) *before* A-to-D quantization. The noise standard deviation of $\sigma_N = 2$ is typical of a number of camera/framegrabber combinations examined in this laboratory. It is not desirable to corrupt the training patterns with noise since a single uncorrupted pattern can be viewed as the mean of a large number of noise corrupted but otherwise identical training patterns, assuming zero mean noise. It is thus computationally more efficient to train with uncorrupted (mean) patterns.

In summary, the synthetic training and validation sets have been constructed using a highly realistic model of the imaging physics and with proper regard to the typical distribution of the different sub-classes of pattern. A similar model has been successfully applied to image feature detection in the past by (Baker et al., 1998; Chen et al., 1996; Rockett, 2003).

3 Genetic Programming Methodology

In this work we have used multiobjective genetic programming to derive an optimal feature extraction stage for edge detection – see Figure 1. Two important caveats should be noted from the outset: Firstly, in this paper we use the terms, "optimal" and "optimization" in the *loose* sense in which they are conventionally used in the evolutionary computing literature, namely to mean near- or approximately optimal. We do not use them in the sense of mathematically optimal since this is an unrealistic expectation for a stochastic optimization method.

Second, the terms "feature extraction", "feature selection" and "feature construction" are variously used in machine learning and related fields, often in contradictory ways. Here we use the term "feature extraction" to mean the (possibly non-linear) combination of (some subset of) raw pattern attributes to form new, more discriminatory features which lie in a decision space. Further, we use the term "feature selection"

to mean the selection of the most discriminatory subset of raw pattern attributes without transformation; feature selection is thus generally a precursor to or component of feature extraction.

The basic multidimensional multiobjective GP methodology employed here has been described previously in (Zhang and Rockett, 2006c). To distinguish it from earlier work in which we used multiobjective GP (MOGP) to project from an p-dimensional pattern space to a 1D decision space (Zhang and Rockett, 2005, 2006b), we term the methodology used here *multidimensional multiobjective GP* (MMOGP). The key advance over (Zhang and Rockett, 2005) is that here we are mapping the p-dimensional pattern vector to an m-dimensional decision space, not just a 1D space; $p = n \times n$ and m is determined as part of the optimization.

Central to the evolution of a p-to-m mapping is the adoption of an appropriate chromosomal structure. Potentially we could evolve m independent feature transforming trees in parallel, the outputs of which could be assembled into an m-dimensional vector. There are, however, practical difficulties in devising a meaningful set of breeding operators for such a scheme. Consequently we have employed a multitree representation similar to (Langdon, 1998) and (Sherrah et al., 1997) – see Figure 3. This multitree approach also allows us to use the set of highly effective genetic operators we have employed previously (Zhang and Rockett, 2006d,c).

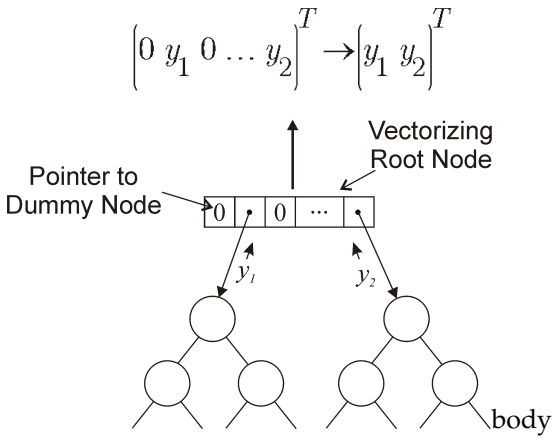

Figure 3: Illustration of the vectorizing tree used in this work; note the role of the root node in assembling the scalar outputs of the m subtrees into an m-dimensional vector.

The initial population was generated with half of the trees of some maximum dimensionality ($m = m_{max}$) and the other half of some randomly chosen dimensionality in the range $m = [1 \dots m_{max}]$. Here we have set $m_{max} = 50$ although in practice all non-dominated Pareto solutions had dimensionalities much less than this figure. All the initial subtrees were generated with random depths in the range [1..7] although subsequently they were allowed to grow without limit, constrained only by a tree complexity objective.

To extend our previous work (Zhang and Rockett, 2005, 2006b) to a multitree chromosome we have added two special node types: *root nodes* and a *dummy nodes*.

As its name implies, a single instance of the root node occurs at the root of the tree. Essentially, its rôle is to assemble the outputs of the m child trees into an m-dimensional decision space vector, each tree supplying one element of the vector. (In terms of implementation, the root node is simply an array of pointers to proper, feature transforming trees). The root node is illustrated in Figure 3.

A dummy node is a special kind of terminal node which can occur anywhere in a tree, including as the immediate child of a root node. Since it is treated as a terminal node, a dummy node can participate fully in the genetic breeding operations although under tree evaluation, a dummy node always returns the value of zero. Thus it contributes no discriminatory power whatsoever to the classification task. Further, during evolution it is the dummy nodes which can alter the dimensionality of the decision space vector. If the child of a root node changes from being a proper subtree to a dummy node then this *reduces* the dimensionality of the decision space vector by one. If the converse happens, this *increases* the decision space vector's dimensionality by one.

Within our overall classification framework, the p-dimensional raw pattern vector is transformed into an m-dimensional decision space vector which forms the input to a Fisher linear discriminant (FLD) classifier. Rather than seek to evolve an integrated feature extractor/classifier which produces class labels as its final output, we evolve a feature extraction stage which is coupled to a conventional classifier. We argue that classifiers are well understood and it makes little sense to devote computational resources to evolving something which already exists; the full computational resource should be used to solve the problem for which no other solution exists, namely feature extraction. We have chosen the FLD as a final classifier because under our methodology, the classifier has to be trained ab initio within the evolutionary loop to determine each individual's fitness. An FLD can be trained rapidly in closed form (Duda et al., 2001) and thus imposes a negligible time burden on the computation. Having projected the set of training patterns into the 1D Fisher direction, we use golden section search to determine the optimal decision threshold.

Our optimization is driven by multiple objectives within a Pareto framework. The first – and most obvious objective – is to minimize the misclassification error (or $0/1$ loss) over the training set. This measure is produced by the trained Fisher linear discriminant classifier set-out above.

It is well-known that unless steps are taken to prevent it, the size of GP trees tends to grow excessively leading to very high computational demands, overfitting and poor generalization; this growth phenomenon is usually termed *bloat*. Thus the second of our objectives is to minimize a measure of tree complexity, very much in sympathy with Occam's Razor where we prefer simpler solutions of equivalent performance. Ekárt and Németh (2001) – among others – have shown that minimizing tree complexity in a multiobjective framework can suppress bloat and we too have used this approach successfully (Zhang and Rockett, 2005, 2006b,c). For an p-to-1 mapping it is convenient to use total node count as a measure of tree complexity but for multitrees in which m is allowed to vary, total node count would impose an unwanted bias in favor of small trees – trees of small m would tend to have fewer nodes in total. Thus, as the second of

our objectives we seek to minimize the mean number of nodes per tree:

$$Complexity = \frac{1}{m}\sum_{i=1}^{m} Size(i)$$

where $Size(i)$ is the node count of the *i*-th sub-tree.

Our third and final objective was motivated by the convergence difficulties we observed in previous work (Zhang and Rockett, 2005). In projecting the pattern vector to a 1D decision space using only the two objectives of $0/1$ loss and the complexity, we observed that the population commonly stagnated. We successfully surmounted this difficulty by adding the third objective of minimizing the Bayes error in the projected 1D decision space. This additional objective proved particularly effective in the initial stages of evolution when the randomly-generated population contained individuals of rather poor performance. The Bayes error appears much more effective than the $0/1$ loss at identifying individuals with slightly greater promise. (We also observed, however, that the Bayes error alone could produce well-trained solutions with very poor generalization performance – see (Zhang and Rockett, 2005) for further details. Thus the Bayes error measure needs to be used in tandem with $0/1$ loss, not in its stead).

In the present work, calculating the Bayes error in an m-dimensional decision space is not practical therefore we have used the PDF overlap in the 1D projected Fisher direction as a measure of class separability. We have calculated this by histogramming the two Fisher-projected, class-conditioned PDFs. Although this final objective is not strictly the Bayes error it has proved to be an easily calculated and effective proxy quantity which exerts selective pressure to separate the two pattern classes.

The evolutionary framework we have employed is a steady-state methodology based on the Pareto converging genetic algorithm (Kumar and Rockett, 2002) which we term *Pareto converging genetic programming* (PCGP). We have found PCGP to give better results than competitor (generational) algorithms across a range of applications, including the present edge detection problem (Zhang and Rockett, 2005, 2006b,a). We thus select two parents for breeding biased in their Pareto dominance, apply crossover and mutation and append the two offspring to the population. After sorting, the two weakest members of the population are discarded. This means that individuals in a PCGP (and PCGA) population can never regress to poorer solutions as can happen in generational schemes. Here we have used a population of 500 and terminated evolution after breeding 20,000 pairs of offspring. (This is roughly equivalent to 100 generations of a generational GA).

The crossover and mutation operators (which are always applied) have been described previously in Zhang and Rockett (2005, 2006d,c). For crossover, we have used the depth-fair method (Ito, et al., 1998) where we select a depth, d in a tree, at which to perform tree splicing; we then select for crossing-over one of the possible sub-trees at this depth, biased in the subtree complexity. That is, we prefer to exchange sub-trees with higher node counts. See Ito et al. (1998) and Zhang and Rockett (2006d) for further details.

In addition, with some probability which we term the *sub-tree preservation crossover probability*, we select a restricted version of crossover which we have found to be very

effective at retaining useful genetic building blocks and speeding convergence (Zhang and Rockett, 2006c). In subtree preservation crossover, rather than choosing a single splicing point in each chromosome tree, we crossover each of the individual subtrees. Thus having selected two parents, A and B for breeding, we perform depth-fair crossover between subtree 1 of parent A and subtree 1 of parent B, then subtree 2 of A and subtree 2 of B, and so on up to subtree q, where $q = min(m_A, m_B)$ and $m_{A,B}$ are the dimensionalities (m-values) of A and B, respectively.

After crossover, both offspring undergo depth-fair mutation in which a subtree is selected on exactly the same lines as for crossover; the selected subtree is then replaced with a new, randomly created subtree. If the root node of a tree is chosen for mutation, the whole tree is replaced by a new random multitree.

The parameters used in this work are shown in Table 1. The "IF-THEN-ELSE" node returns the value of the second child if the first child value is greater than zero, otherwise the third child value is returned. The "MAX" node returns the larger value from its two successors while "MIN" returns the smaller. If both successor node values are larger than zero, "XOR" returns zero, otherwise it returns unity. "AND" returns the boolean result from its two operands, returning the value of unity only when both are non-zero; in the same way, "OR" returns unity if at least one operands is non-zero.

Terminal set	Input pixel values from 13 × 13 image patches
	Dummy nodes
	10 floating point numbers ∈ {0...1}
Function set	SQRT, LOG, POW2, UNARY MINUS,SIN, NOT
	-, +, , /, MAX, MIN, XOR, OR, AND
	IF-THEN-ELSE (IFTE)
Population size	500
Initial population	Half full trees, half random trees
Initial max. tree depth	7
Sub-tree preservation probability	0.2
Max. no. of breeding cycles	20,000
Stopping criterion	Max. generations exceeded

Table 1: MMOGP (PCGP) settings used in this work

4 Results

Taking the training set described in Section 2 and the multidimensional multiobjective GP methodology of Section 3 we evolved a set of edge detectors, each non-dominated individual on the Pareto front comprising a possible solution which trades off complexity, 0/1 loss over the training set and the Bayes error-like measure of class separation. We have run the algorithm 10 times with different initial populations and obtained rather similar solutions each time.

The set of Pareto front solutions from a single typical run is shown in Figure 4 in which we plot misclassification error against the total number of nodes in the candidate solutions. We have plotted total node number rather than the mean subtree complexity

measure which we actually seek to minimize since this gives a more accurate idea of the range of solution complexities. (Recall from Section 3 that we only employed the mean node measure so as not to bias our objective towards generating trees of small dimensionality.) In Figure 4 a number of individuals with total node counts in the range 10-15 appear to be dominated; in fact, Figure 4 is a two-dimensional *projection* of a three-dimensional Pareto front – we minimize three simultaneous objectives. When the third (unshown) objective is accounted for, the proper dominance relations are observed; a plot of this 3D space is uninformative and hence not shown here. In addition, we actually minimize mean node count rather than total node count and taking this extra factor into account, the solutions in Figure 4 are indeed non-dominated in the 3D objective space.

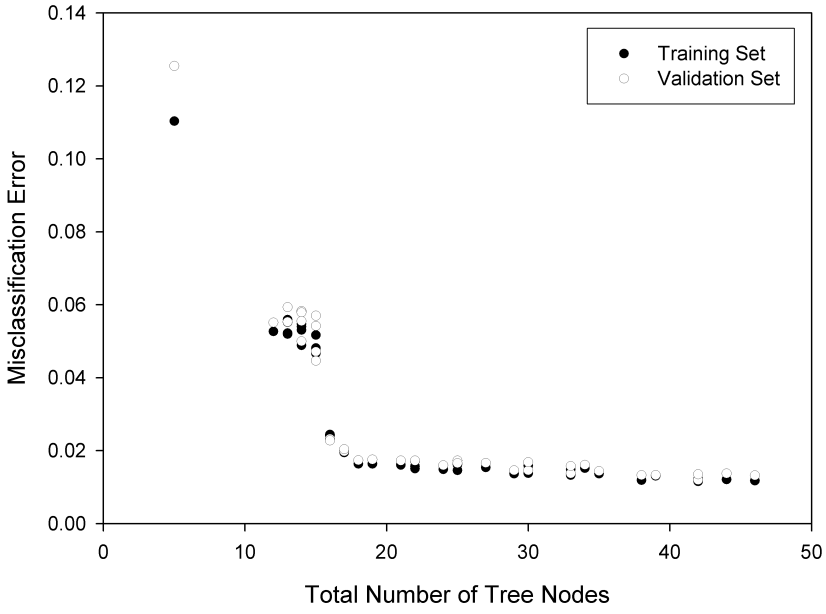

Figure 4: Typical two-dimensional projection of the 3D Pareto front showing both training set and (independent) validation set errors; see text for details.

From Figure 4 we can see that the misclassification errors fall to a roughly constant value for trees with more than 20 nodes although there is a further small decrease in error with increasing tree complexity. Reassuringly, the training set and validation set errors (shown as filled and open circles, respectively) are fairly similar implying that the evolved detectors are generalizing quite well.

(The results in Figure 4 have been produced with a population size of 500; repeating the evolution with a population of 1000 produced no better solutions.)

Although the multiple objectives play a pivotal role in shaping the solutions, in classification problems we are ultimately concerned with obtaining the lowest possible misclassification error (0/1 loss). In particular, in this problem we have available a

large (independent) validation set and so we have selected the solution with the lowest validation error for further study. In fact, we observed that there was a cluster of solutions which all exhibited very similar validation errors around the minimum, all of which had dimensionalities (m-values) of 9 or 10. Our final suggested 9-dimensional solution is shown in Figure 5. We make no attempts to explain or justify the sequence of evolved operations. It is widely accepted in the evolutionary computing community that interpreting GP trees is usually very difficult or impossible. Other than the phenomenological observation that this sequence of processing steps works and – as we show below – produces superior quantitative results, we can offer no justification for our solution. The largely phenomenological nature of genetic programming is arguably a major reservation about the technique: it provides effective solutions without giving insight. That said, it is interesting that the tree in Figure 5 uses so many raw, untransformed pixel values and in fact, does comparatively little processing, as is apparent from the explicit equations set-out in the Appendix. We make no claims about the *global* optimality of this detector, just that it performs significantly better than the benchmark Canny detector and our previous 1D-MOGP edge detector: it is the best performing edge detector we have found so far.

(A second, although quite unintended benefit of the simplicity of the final detector is that it is likely to be quite fast in operation although because this was never a design objective, we have not investigated this aspect of performance.)

One of the beneficial side-effects of evolution under parsimony pressure is that we obtain feature selection as part of the process of minimizing tree complexity and therefore the number of terminals. The suggested detector's utilization of raw pixels over the 13×13 image patch is shown in Figure 6 from which it can be seen that, apart from an intuitively pleasing use of pixels around the center of the image patch where one would expect the greatest amount of edge discriminating information to be found, there appears to be an asymmetric distribution of selected pixels with a pronounced NW-SE bias. This asymmetry is somewhat at odds with our prior notion of an edge having no preferred orientation and leads us to speculate that we could improve performance by constraining the evolution to symmetrical groupings of pixels. Embedding a priori knowledge in GP could further improve the outcome although there is always the danger of carrying this process to far. This remains an area for future research.

The validation error of our best performing tree is shown in Table 2 together with the performance of the Canny algorithm (with and without non-maximal suppression) and the 1D-MOGP detector reported in (Zhang & Rockett, 2005b). In Table 2 neither of the multiobjective GP techniques has employed NMS.

It is clear from Table 2 that the present MMOGP technique yields the best performance followed by the 1 D-MOGP method (Zhang & Rockett, 2005b) with Canny yielding the worst, even after NMS postprocessing; the Canny detector was operated at its point of minimum Bayes risk (Zhang and Rockett, 2006e). The differences between these results are statistically significant to at least the 99% confidence level.

In addition to obtaining results on a physically realistic synthetic dataset, we also report both quantitative and qualitative results on the hand-labeled USF test

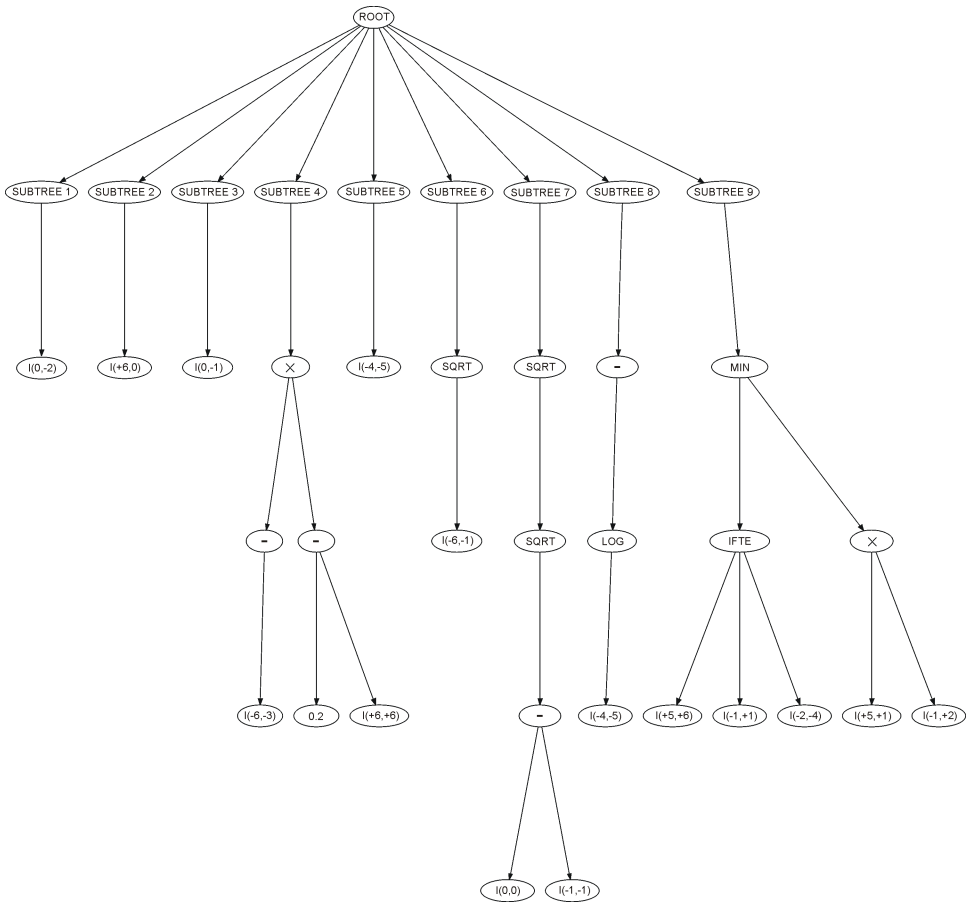

Figure 5: The final suggested edge detecting tree

images (Bowyer et al., 2001). The use of these images requires a caveat since we have previously shown (Zhang and Rockett, 2006e) that the censoring of the data together with labeling errors can produce some inconsistent results. Nonetheless, we include these results for the sake of completeness.

Since postprocessing stages like non-maximal suppression (NMS) can significantly improve the performance of the Canny algorithm, we have also devised an NMS post-processing scheme for our GP-generated detectors. NMS can thin edge maps to single pixel width lines by reducing the false positive labelings (Zhang and Rockett, 2006e). Our GP detectors – both MOGP and MMOGP – classify edges with respect to a threshold in the projected Fisher direction; we take the 'distance' of an individual edgel's response from this threshold as a measure of edge response. The greater the distance from the threshold value, the more certain we are about the correctness of the assigned label. Using a conventional image processing difference-of-boxes operator, we estimate the orientation of the edge which we quantize into one of eight directions. We then examine the edge response of the central pixel and the two pixels on either side of this central pixel (approximately) normal to the edge; the response of the central pixel is

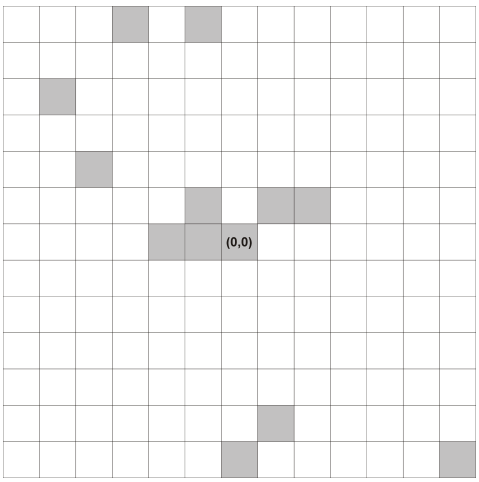

Figure 6: Pixel utilization across the 13 $\times$ 13 image patch in the final suggested edge detecting tree

Method	**Classification Error**
MMOGP w/o NMS	0.012
1D GP w/o NMS (Zhang & Rockett, 2005b)	0.0264
Canny w/o NMS	0.0496
Canny with NMS	0.0436

Table 2: Classification error comparisons for Canny, 1D-MOGP (Zhang and Rockett, 2005) and the present MMOGP methods.

set to zero if its value is not the largest of the three examined responses (Jain et al., 1995).

The image labeling results from the USF data are shown in Figures 7 to 10. In this sequence of figures, (a) is the original image, (b) is the labeled ground truth image, (c) is the Canny edge map, (d) is the 1D-MOGP edge map, (e) is the MMOGP edge map, (f) is the Canny result + NMS, (g) is the 1D-MOGP result + NMS and (h) is the MMOGP result + NMS. The Canny threshold (both with and without NMS) were determined independently for each image by minimizing the Bayes risk under the assumption of equal costs for false positives and true negatives; we have no reason to suppose that one sort of error is any more important than another and so we take a neutral stance on the relative costs of errors. On the other hand, the labeling thresholds in both the GP detectors were fixed by the evolutionary learning procedure under the assumptions of a constant 5% edge prior.

From comparing the raw edge maps (c), (d) and (e) in Figures 7 to 10, a number of points become apparent. Firstly, the Canny edge maps (c) are very sparse since this detector has been shown to be poor without postprocessing stages (Zhang and Rockett, 2006e). Comparing the edge maps of 1D-MOGP detector (Zhang and Rockett, 2005, 2006b) in (d) with the present MMOGP method in (e), it is clear that the MMOGP algorithm labels more perceptual edges and fewer false positives.

Quantitatively, this observation is confirmed in Tables 3 to 5. Thus in terms of the raw edge maps, the results on the USF imagery confirm the conclusions drawn from Table 2.

Figure	Edge Prior	Without NMS		With NMS	
		TP	FP	TP	FP
7	0.087	0.0003	0.0001	0.3821	0.0466
8	0.080	0.0204	0.0003	0.4326	0.0695
9	0.211	0.0142	0.0063	0.3886	0.0061
10	0.066	0.0048	0.0003	0.3580	0.0049

Table 3: Canny [TP, FP] operating points for the USF test images

Figure	Edge Prior	Without NMS		With NMS	
		TP	FP	TP	FP
7	0.087	0.5045	0.0278	0.3657	0.0083
8	0.080	0.4151	0.0298	0.3388	0.0052
9	0.211	0.6228	0.0129	0.4433	0.00036
10	0.066	0.5581	0.0246	0.3415	0.0047

Table 4: 1D-MOGP [TP, FP] operating points for the USF test images

Figure	Edge Prior	Without NMS		With NMS	
		TP	FP	TP	FP
7	0.087	0.7358	0.0215	0.6561	0.0090
8	0.080	0.7219	0.0151	0.6617	0.0027
9	0.211	0.7727	0.0110	0.6674	0.0013
10	0.066	0.6496	0.0175	0.3801	0.0005

Table 5: MMOGP [TP, FP] operating points for the USF test images

Comparisons of the labeling results after NMS are shown in the (f), (g) and (h) sub-figures. Clearly the Canny results display significant numbers of false positives. After NMS, the MMOGP detector also shows obviously fewer false positives in Figure 7 and a somewhat smaller false positive advantage in Figure 8 compared to the 1D-MOGP approach. In the rather cluttered scenes in Figures 9 and 10, MMOGP yields rather more false positives than 1D-MOGP. On the other hand, MMOGP consistently attains much better true positive rates across all four images. Again, these data are summarized in Tables 3 – 5. It is noteworthy that the stapler image in Figure 10 proves somewhat more problematic than the others although the reason for this is not clear. Possibly the number of labeling errors is particularly high for this image.

The Bayes risk values for all three detectors, with and without NMS are shown in Table 6. For the case of the Canny detector, the risk values are the lowest which can be attained by adjusting the labeling threshold. It can be seen that MMOGP returns the lowest risk across all four images, both without or with NMS. For the two GP-based

detectors, NMS actually makes the risk worse for the drinking fountain image (Figure 9). The Bayes risk is a weighted sum of the risk of true negatives (missed edges) and the risk of false positives (mislabeled non-edges), where the relative weightings are determined by the edge prior. This increase in risk is due to the drinking fountain image having a rather high prior of 0.221 whereas both GP detectors were trained assuming an edge prior of 0.05. NMS will decrease the false positive rate but increase the true negative rate; due to the effective weightings on these terms, the overall Bayes risk increases after NMS.

Figure	**Without NMS**			**With NMS**		
	MMOGP	**1D-MOGP**	**Canny**	**MMOGP**	**1D-MOGP**	**Canny**
7	0.06054	0.06848	0.0871	0.03825	0.06276	0.09630
8	0.03612	0.07420	0.0786	0.02960	0.05768	0.10933
9	0.05664	0.08977	0.2129	0.07120	0.11774	0.13381
10	0.03951	0.05214	0.06596	0.04173	0.04785	0.04694

Table 6: Bayes risk comparisons for the USF test images

5 Conclusions and Future Work

In this paper we have reported the evolution of a feature extraction stage for edge detection using multidimensional multiobjective genetic programming. Employing a realistic model of the imaging physics, we have generated both training and validation datasets and used these to evolve a p-to-m mapping ($p = n \times n$) which projects the pixel intensities of the image patch into an m-dimensional decision space; the value of m is also optimized in this process. Within a multiobjective Pareto framework we have minimized: a Bayes error-like measure which has proved successful in speeding convergence, the misclassification error over the training set and a measure of tree complexity.

The suggested edge detector has been shown by means of an independent validation set to provide superior performance to both the conventional Canny detector and to earlier multiobjective GP work which projected the pattern vector into a 1D decision space (Zhang and Rockett, 2005). In addition, we have used the hand-labeled USF image set to demonstrate the effectiveness of the proposed edge detector on real imagery.

Feature selection is obtained as a beneficial side-effect in this work since we have (implicitly) minimized the number of terminal (pixel) values used. Nonetheless, the final detector's selection of pixels away from the central cell is rather asymmetric and therefore rather confounds our initial assumption that edges have no particular orientation; we might anticipate further improvements in performance if symmetry can be imposed on the receptive field of pixels. This is an area for future work.

Our image feature detection methodology is not restricted to edges. Indeed Baker et al. (1998) have reported using a similar data model to detect a range of image features, albeit in a non-evolutionary approach; Chen and Rockett (1997) have also reported training neural network corner detectors with this type of data model. It

is thus an area of future work to apply the present methodology to evolving corner detectors. More generally, geometric feature detectors (Baker et al., 1998) across a range of imaging modalities are amenable to this approach.

Appendix

The edge detector proposed here takes the raw pixel intensities from a 13 $\times$ 13 patch to form a 9-dimensional decision space vector, $\mathbf{y}$. Denoting the pixel intensities as $I_{r,c}$ where $r, c \in [-6... + 6]$ and the central cell of the 13 $\times$ 13 patch is given the row/column indices of $(0, 0)$, the individual elements of $\mathbf{y}$ are given by:

$$\begin{aligned}
y_1 &= I_{0,-2} \\
y_2 &= I_{+6,0} \\
y_3 &= I_{0,-1} \\
y_4 &= -I_{-6,-3} \times [0.2 - I_{+6,+6}] \\
y_5 &= I_{-4,-5} \\
y_6 &= \sqrt{I_{-6,-1}} \\
y_7 &= [I_{0,0} - I_{-1,-1}]^{1/4} \\
y_8 &= -\log(I_{-4,-5}) \\
y_9 &= \min(a, I_{+5,+1} \times I_{-1,+2})
\end{aligned}$$

where:

$$a = \begin{cases} I_{-1,+1} & I_{+5,+6} > 0 \\ I_{-2,-4} & I_{+5,+6} \leq 0 \end{cases}$$

The output of the 9-dimensional tree, $\mathbf{y}$ is multiplied by the Fisher projection vector, $\mathbf{f}$ such that the scalar value in the projection direction is given by:

$$v = \mathbf{f}.\mathbf{y}$$

and where the elements of $\mathbf{f}$: $f_i, i \in [1...9]$ are given in Table 7.

Element Index	Element Value
1	-1.2872555339074174 $\times 10^{-6}$
2	-1.3624975402259024 $\times 10^{-8}$
3	4.5661303927836337 $\times 10^{-7}$
4	-3.9071898753929782 $\times 10^{-9}$
5	9.7621791371689665 $\times 10^{-7}$
6	-1.7820932457027433 $\times 10^{-5}$
7	-0.0011925937292865041
8	-1.6771130738254199 $\times 10^{-5}$
9	1.4390194474984800 $\times 10^{-6}$

Table 7: Values of the elements of the Fisher projection vector

Finally, a decision threshold was applied such that if $v \geq 0.0018490142574468711$ then the central pixel of the image patch is adjudged to be an edge.

References

Abdou, I. E. and Pratt, W. K. (1979). Quantitative design and evaluation of enhancement/thresholding edge detectors. *Proceedings of the IEEE*, 67(5):753–763.

Baker, S., Nayar, S. K., and Murase, H. (1998). Parametric feature detection. *International Journal of Computer Vision*, 27(1):27–50.

Bleuler, S., Brack, M., Theile, L., and Zitzler, E. (2001). Multiobjective genetic programming: Reducing bloat using SPEA2. In *Congress on Evolutionary Computation*, pages 536–543, Seoul, Korea. IEEE.

Born, M. and Wolf, E. (1999). *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Cambridge University Press, Cambridge.

Bowyer, K. W., Kranenburg, C., and Dougherty, S. (2001). Edge detector evaluation using empirical ROC curves. *Computer Vision and Image Understanding*, 84(1):77–103.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698.

Chen, W.-C. and Rockett, P. I. (1997). Bayesian labelling of corners using a grey-level corner image model. In *Proceedings of IEEE International Conference on Image Processing (ICIP'97)*, pages 687–690, Santa Barbara, CA. IEEE.

Chen, W.-C., Thacker, N. A., and Rockett, P. I. (1996). An adaptive step edge model for self-consistent training of a neural network for probabilistic edge labelling. *IEE Proceedings – Vision, Image and Signal Processing*, 143(1):41–50.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Recognition*. John Wiley and Sons, New York.

Ekárt, A. and Németh, S. Z. (2001). Selection based on the Pareto nondomination criterion for controlling code growth in genetic programming. *Genetic Programming and Evolvable Machines*, 2(1):61–73.

Harris, C. and Buxton, B. (1996). Evolving edge detectors with genetic programming. In *1st Annual Conference on Genetic Programming*, pages 309–314.

Heath, M. D., Sarkar, S., Sanocki, T., and Bowyer, K. W. (1998). Comparison of edge detectors: A methodology and initial study. *Computer Vision and Image Understanding*, 69(1):38–54.

Ito, T., Iba, H., and Sato, S. (1998). Non-destructive depth-dependent crossover for genetic programming. In *1st European Workshop on Genetic Programming*, pages 14–15, Paris, France.

Jain, R. C., Kasturi, R., and Schunk, B. G. (1995). *Introduction to Machine Vision*. McGraw-Hill.

Jalali, S. and Boyce, J. F. (1995). Determination of optimal general edge detectors by global minimization of a cost function. *Image and Vision Computing*, 13(9):683–693.

Kumar, R. and Rockett, P. I. (2002). Improved sampling of the Pareto-front in multiobjective genetic optimizations by steady-state evolution: A Pareto converging genetic algorithm. *Evolutionary Computation*, 10(3):283–314.

Langdon, W. (1998). *Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming!* Kluwer Series on Genetic Programming. Kluwer, Boston, MA.

Lyvers, E. P. and Mitchell, O. R. (1988). Precision edge contrast and orientation estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):927–937.

Petrou, M. and Bosdogianni, P. (1999). *Image Processing: The Fundamentals*. John Wiley and Sons, Chichester, UK.

Rockett, P. I. (2003). Performance assessment of feature detection algorithms: A methodology and case study on corner detectors. *IEEE Transactions on Image Processing*, 12(11):1668–1676.

Sherrah, J. R., Bogner, R. E., and Bouzerdoum, A. (1997). The evolutionary pre-processor: Automatic feature extraction for supervised classification using genetic programming. In *2nd Annual Conference on Genetic Programming*, pages 304–312, Palo Alto, CA.

Tsai, A., Yezzi, A., and A.S.Willsky (2001). Curve evolution implementation of the Mumford-Shah functional for image segmentation, denoising, interpolation and magnification. *IEEE Transactions on Image Processing*, 10(8):1169–1186.

Zhang, Y. and Rockett, P. (2005). 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)*, pages 795–802, Washington, DC. ACM Press.

Zhang, Y. and Rockett, P. I. (2006a). Comparison of evolutionary strategies for multi-objective genetic programming. In *IEEE Systems, Man Cybernetics Society Conference on Advances in Cybernetic Systems (AICS2006)*, Sheffield, UK.

Zhang, Y. and Rockett, P. I. (2006b). Feature extraction using multi-objective genetic programming. In Jin, Y., editor, *Multi-Objective Machine Learning*. Springer, Heidelberg.

Zhang, Y. and Rockett, P. I. (2006c). A generic multi-dimensional feature extraction method using multiobjective genetic programming. *Submitted to Evolutionary Computation*.

Zhang, Y. and Rockett, P. I. (2006d). A generic optimal feature extraction method using multiobjective genetic programming: Methodology and applications. *Submitted to IEEE Transactions on Knowledge and Data Engineering*.

Zhang, Y. and Rockett, P. I. (2006e). The Bayesian operating point of the Canny edge detector. *IEEE Transactions on Image Processing*, 15(11):3409– 3416.

Zheng, S., Liu, J., and Tian, W. (2004). A new efficient SVM-based edge detection method. *Pattern Recognition Letters*, 25(10):1143–1154.

Zhou, Y. T., Venkateshwar, V., and Chellappa, R. (1989). Edge detection and linear feature extraction using a 2D random field model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(1):84–95.

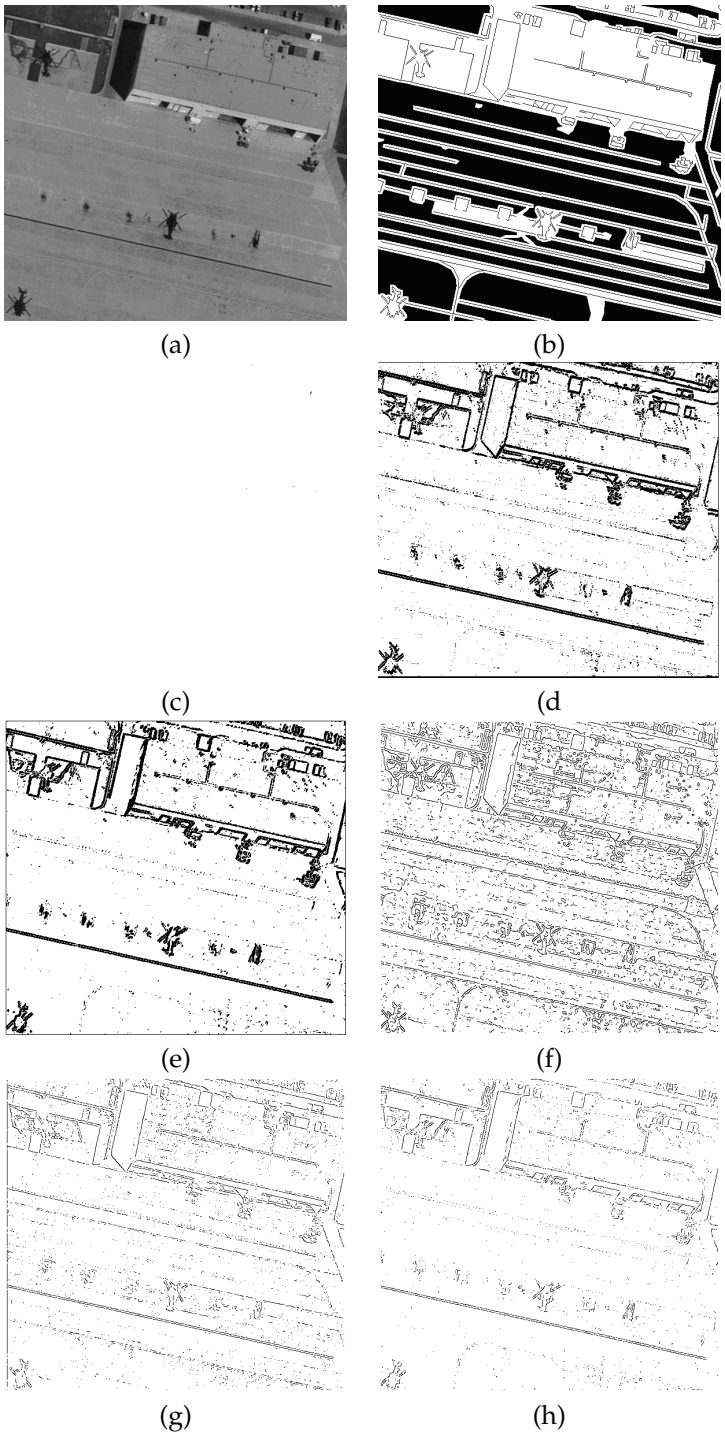

Figure 7: (a) to (h) illustrate the comparisons from the Canny edge detector and 1D-MOGP and MMOGP.

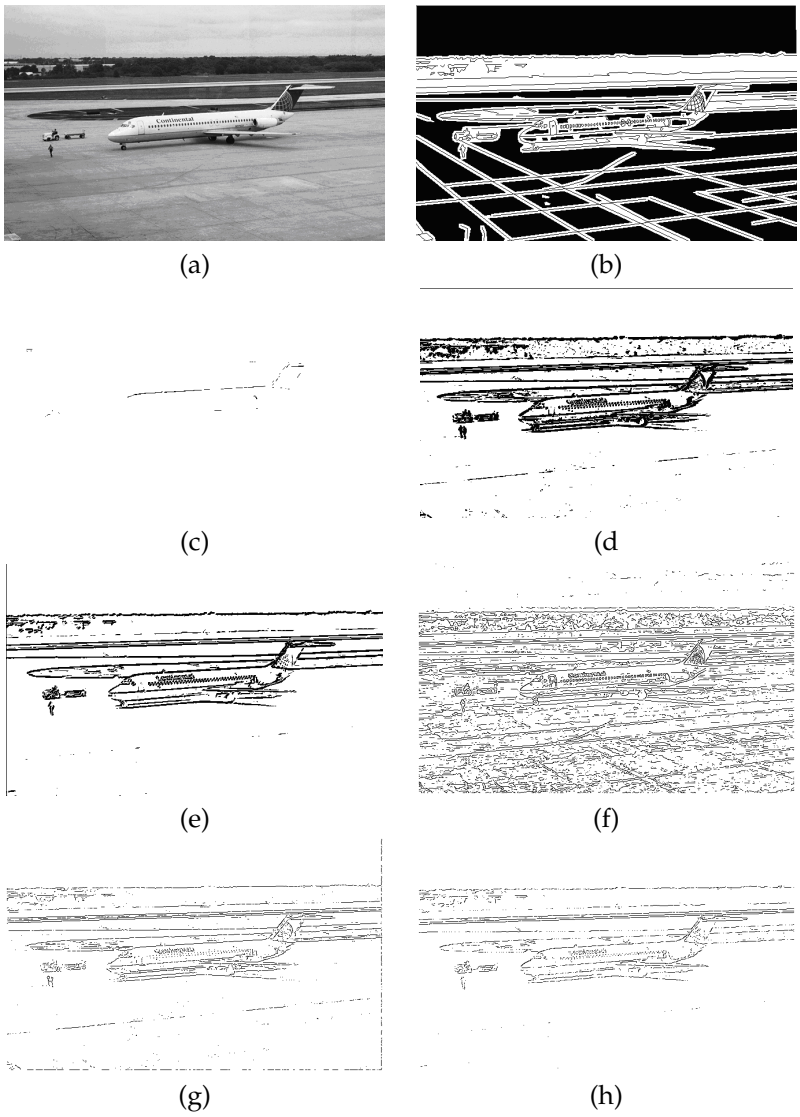

Figure 8: (a) to (h) illustrate the comparisons from the Canny edge detector and 1D-MOGP and MMOGP

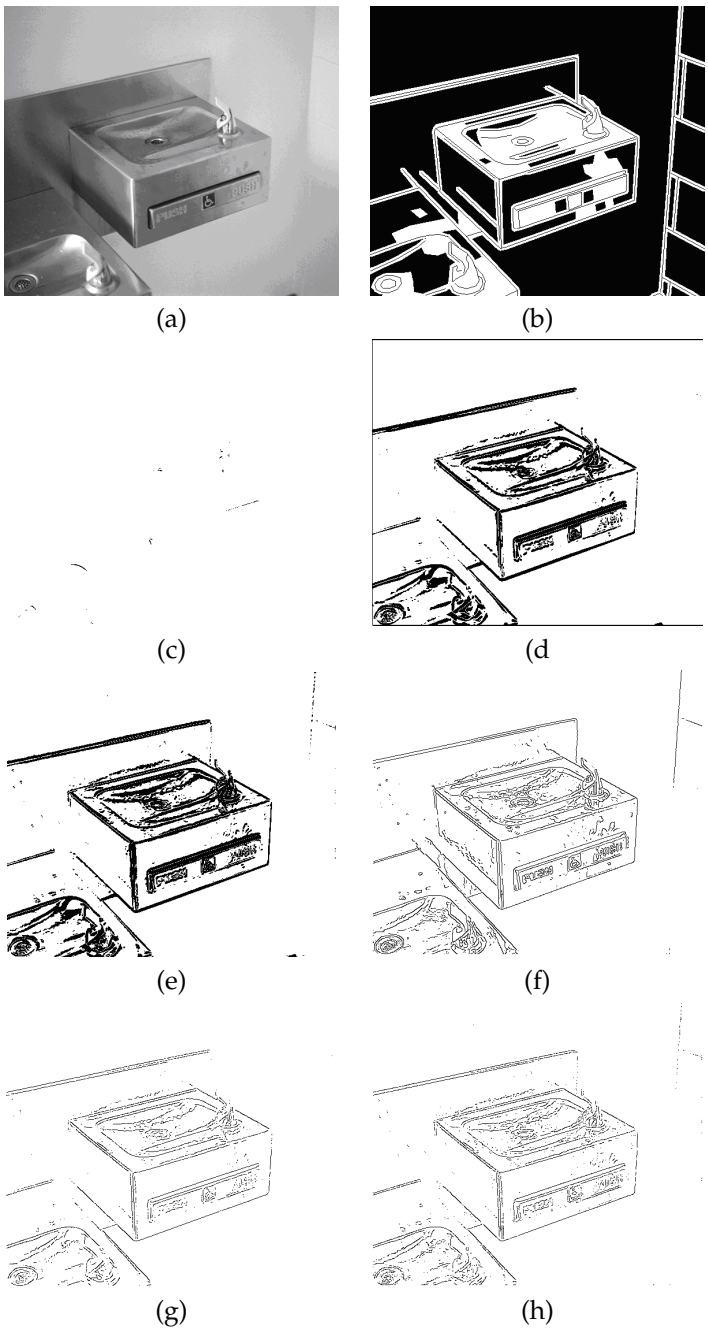

Figure 9: (a) to (h) illustrate the comparisons from the Canny edge detector and 1D-MOGP and MMOGP

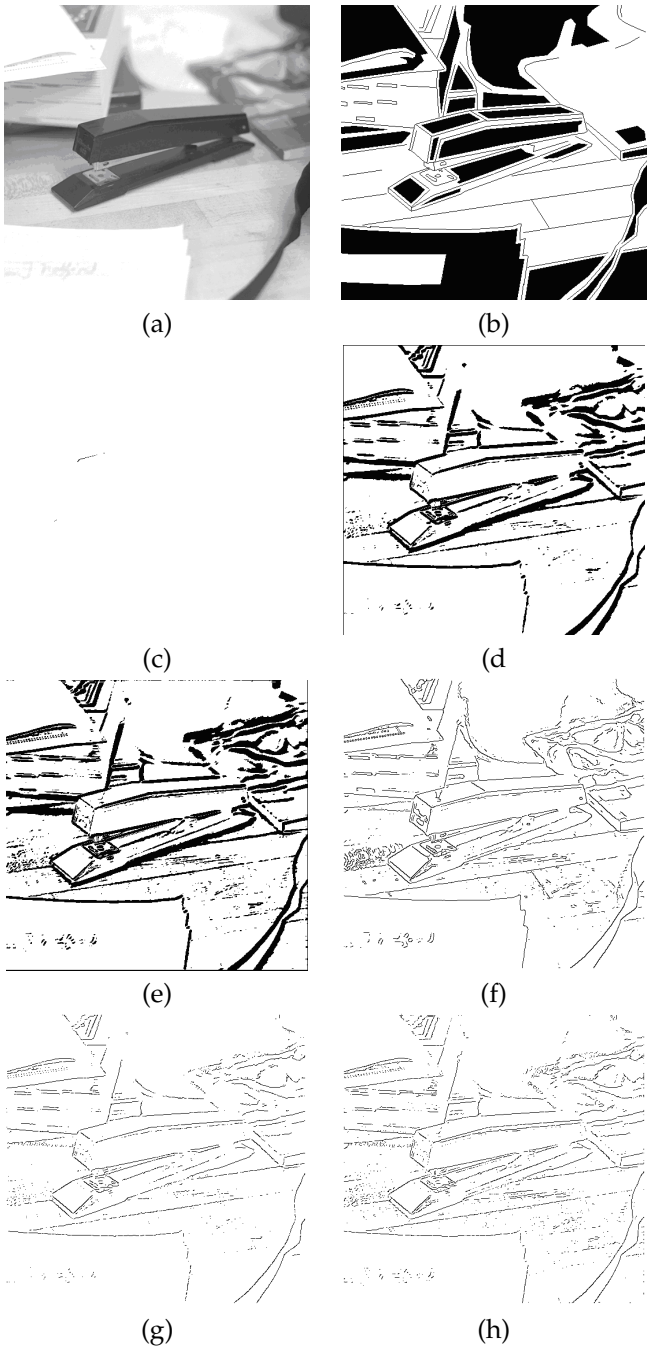

Figure 10: (a) to (h) illustrate the comparisons from the Canny edge detector and 1D-MOGP and MMOGP