Noname manuscript No.
(will be inserted by the editor)

Bayesian Genetic Programming for Edge Detection

Wenlong Fu · Mengjie Zhang · Mark Johnston

Received: date / Accepted: date

Abstract In edge detection, designing new techniques to combine local features is expected to improve detection performance. However, how to effectively design combination techniques remains an open issue. In this study, an automatic design approach is proposed to combine local edge features using Bayesian programs (models) evolved by Genetic Programming (GP). Multivariate density is used to estimate prior probabilities for edge points and non-edge points. Bayesian programs evolved by GP are used to construct composite features after estimating the relevant multivariate density. The results show that GP has the ability to effectively evolve Bayesian programs. These evolved programs have higher detection accuracy than the combination of local features by directly using the multivariate density (of these local features) in a simple Bayesian model. From evolved Bayesian programs, the proposed GP system has potential to effectively select features to construct Bayesian programs for performance improvement.

Keywords Genetic Programming · Edge Detection · Bayesian Model · Feature Construction

1 Introduction

In order to extract information from an object, an essential job is to distinguish the object from its background. Edge detection is a process of detecting discontinuous changes between objects and background in an image [2, 37]. In general, the edges extracted from an image can be simplified to describe the image content. Edge detection benefits a wide range of applications, such as image compression [33], object tracking [35], and image retrieval [23]. In a grayscale image, an edge feature for each pixel is a general mathematical formula of intensity values of pixels in a local area, and it is employed in the process of marking the pixel as an edge point or a non-edge point. To extract edge features, many methods have been proposed [30, 15, 3]. From human observations, several different solutions might be considered as true edge maps of one natural image. Therefore, edge detection is a subjective task. Different local edge features have been combined to improve detection performance [28, 9].

Our previous work [11] has shown that Genetic Programming (GP) can be successfully applied to feature construction by combining basic features and estimating the distribution of the observations of evolved programs. The basic edge features come from different prior domain knowledge (the Gaussian gradient, the normalised standard deviation, and the histogram gradient). The constructed GP edge features were significantly better than the combination from a simple Bayesian model using a general multivariate normal density [7].

From machine learning [7], different methods, such as Bayesian techniques, can be applied for combining basic features. Bayesian techniques have been widely applied to classification [21]. Since these techniques are based on applying Bayes' theorem [7], human experts can understand the models using the Bayesian techniques. However, there are still existing issues in the Bayesian techniques, such as how to effectively select features and design Bayesian models [21]. It is there-

Wenlong Fu. and Mengjie Zhang
School of Engineering and Computer Science, Victoria University of Wellington, PO Box 600, Wellington, New Zealand
Tel.: +64-4-4635233-7059
Fax: +64-4-4635045
E-mail: Wenlong.Fu@gmail.com

Mark Johnston
University of Worcester, United Kingdom.

fore desirable to develop a GP system to automatically select features and design Bayesian models for performance improvement of a Bayesian technique.

The existing work using GP in edge detection mainly focuses on the construction of low-level edge detectors, such as approximating one-dimen-sional filters [31], selecting raw pixels [38], and combining image operators [22]. The work using GP in edge detection for the construction of low-level edge detectors shows that GP has potential to effectively evolve good edge detectors [38, 22, 10]. We have conducted initial investigation of employing GP to evolve Bayesian programs [14]. The evolved programs have higher detection accuracy than a simple Bayesian model to combine a set of predefined basic edge features.

1.1 Goal

The goal of this paper is to investigate automatic high-level feature construction for edge detection using GP to evolve Bayesian programs. Here, a high-level feature means that it is constructed based on some basic features and combination techniques with prior domain knowledge. In our proposed Bayesian GP system [14], a simple Bayesian model with a general multivariate normal density was proposed as a function to construct different Bayesian programs. Different from our previous work [14], this paper further investigate the Bayesian GP system. The simple Bayesian model can be used as a terminal as well. How to effectively use the Bayesian model with a general multivariate normal density in the GP system is investigated. Specifically, the following research objectives will be investigated.

- Whether there are differences between using all basic features and randomly selecting a set of features in a Bayesian function.
- Whether using the simple Bayesian model as a terminal can achieve better performance than using the Bayesian model as a function.
- Whether the evolved Bayesian programs can be reasonably interpreted.

1.2 Organisation

In the remainder of the paper, Section 2 describes relevant background on edge detection and using GP for edge detection. Section 3 introduces a Bayesian GP system further developed from our previous work [14] to construct high-level features. Section 4 presents the design of the experiments. After giving the results in Section 5 with discussions, Section 6 provides further discussions. Finally, Section 7 draws conclusions and suggests future work directions.

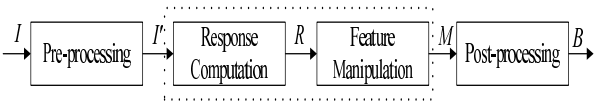


Fig. 1 General edge detection flow.

2 Background

This section gives background on edge detection and discusses the existing work using GP for edge detection.

2.1 Edge Detection

Typically, edge detection includes three stages: pre-processing, feature extraction and post-processing [30]. A general process flow for edge detection is shown in Fig. 1. Given an image I, pre-processing techniques will use I as input and generate an intermediate result I'. In the pre-processing stage, noise and textures will usually be suppressed with little influence on edges. The second stage is to extract edge features, including computing responses on pixels (feature values) and manipulating features. In the process of response computation, edge responses, such as image derivatives [5], are employed. Based on different directional derivatives, a set of features R is obtained. Note that it is possible to combine the pre-processing stage and the response computation stage together. For example, image Gaussian gradients include image gradients and Gaussian filtering [5]. In the process of manipulating edge features, feature selection [6] and further feature construction [17, 28] might be used to generate a set of features M. In edge detection, the process of extracting edge features is perhaps the most important. In the post-processing stage, marking edge points, thinning edges, linking broken edge points, and deleting stand-alone edge points are usually used. A final binary edge map B is obtained after finishing post-processing operations. Generally, pre-processing and post-processing can be routinely applied to different edge feature extraction techniques.

To measure an edge feature based on ground truth, the feature is used to obtain final binary edge maps after applying thresholding techniques and post-processing techniques. For example, the F-measure technique [28] has been used to evaluate edge detectors and boundary detectors. Since common post-processing techniques, such as thinning operations [26] and non-maximum suppression [5], can generally follow most feature extraction approaches, it is also important to evaluate the features

without non-maximum suppression and other specific post-processing techniques [29].

2.2 Local Features

A local moving window is usually used to extract edge features for each pixel. Local features are usually based on moving windows. The local rotation-invariant features used in this paper are introduced as follows.

2.2.1 Image Gaussian Gradient

Differentiation-based techniques are popular in edge feature extraction. For example, the image derivative [5, 17] and the local histogram derivative [28] have been used for extracting edge features. Horizontal derivatives of an image are used to detect vertical edges in the image. For example, to use a feature to detect vertical edges, the horizontal derivative of a Gaussian filter is defined in Equation (1), where σ is the parameter scale of the Gaussian filter, and (u, v) is the position of a neighbour relative to a centre discriminated pixel. Equation (2) describes the vertical derivative of a Gaussian filter. Given a direction $\theta = 0^\circ$ or 90°, the relative image derivative $T_{gg,\theta}(x, y)$ is obtained after the convolution of image I is performed with the derivative $g_\theta(u, v)$. Equation (3) describes the image Gaussian derivative $T_{gg,\theta}(x, y)$. Equation (4) defines the image Gaussian gradient T_{gg}. Here, T_{gg} combines $T_{gg,0^\circ}(x, y)$ and $T_{gg,90^\circ}(x, y)$ as a rotation-invariant feature.

$$g_{0^\circ}(u, v) = -\frac{u}{2\pi\sigma^4} \exp\left(-\frac{u^2 + v^2}{2\sigma^2}\right) \tag{1}$$

$$g_{90^\circ}(u, v) = -\frac{v}{2\pi\sigma^4} \exp\left(-\frac{u^2 + v^2}{2\sigma^2}\right) \tag{2}$$

$$T_{gg,\theta}(x, y) = g_\theta(u, v) \circledast I(x, y) \tag{3}$$

$$T_{gg}(x, y) = \sqrt{T^2_{gg,0^\circ}(x, y) + T^2_{gg,90^\circ}(x, y)} \tag{4}$$

2.2.2 Image Histogram Gradient

Objects normally have different histograms. Image local histogram derivatives have been applied to edge detection [28]. To detect pixel (x, y) as an edge point at direction θ, the image local histogram derivative around this pixel is defined in Equation (5). Here, a separate line with direction θ over pixel (x, y) is used to divide the pixel's neighbours in a local area into the left and right groups. The values of these neighbours are binned. In bin i, $l_{\theta,i}$ is the number of occurrences of the neighbours in the left group, and $r_{\theta,i}$ is the number of occurrences of the neighbours in the right group. The local histogram gradient T_{hg} is defined in Equation (6). Note that $h_\theta(x, y) \geq 0$ and T_{hg} is the sum of $h_\theta(x, y)$ in a set of possible directions.

$$h_\theta(x, y) = \tfrac{1}{2} \sum \frac{(l_{\theta,i} - r_{\theta,i})^2}{l_{\theta,i} + r_{\theta,i}} \tag{5}$$

$$T_{hg} = \sum_\theta h_\theta(x, y) \tag{6}$$

2.2.3 Normalised Standard Deviation

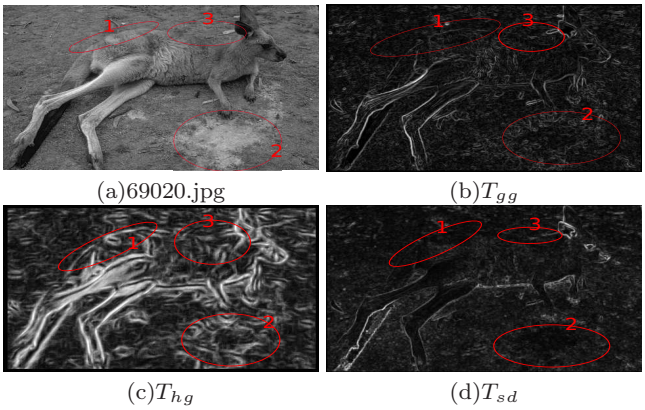
In the literature, the normalised standard deviation is seldom employed as a feature for edge detection. Generally, it is used as an the indication of the image quality [32]. Equation (7) describes the local normalised standard deviation $T_{sd}(x, y)$. Here $Mean(x, y)$ is the mean of the pixel intensities in a local area around pixel (x, y), and $SD(x, y)$ are the standard deviation of the pixel intensities in the local area. The local area is typically a small moving window. Note that image rotation does not affect $T_{sd}(x, y)$ because the local area is rotation-invariant.

$$T_{sd}(x, y) = \frac{SD(x, y)}{Mean(x, y)} \tag{7}$$

The notion of edges is the result of common human experience rather than a formal mathematical definition [30, 27]. The local features from derivatives include high responses on non-edge points which are affected by noise or textures. For instance, Fig. 2 shows the three different edge features T_{gg}, T_{hg} and T_{sd} applied to an image from the Berkeley Segmentation Dataset (BSD) [28]. Fig. 2 (b) has problems in area 1 where it gives inappropriately strong responses on non-edge points and weak responses on edge points, area 2 where there is a lot of noise interference, and area 3 where edge points are not detected. Fig. 2 (c) shows better responses on the edges in area 1 than Fig. 2 (b), but has a problem in area 2 of noise interference, and only includes a few edge points in area 3 with many false alarms. Fig. 2 (d) also has a problem in area 1 where it gives too weak responses to the edge points, but is only weakly affected by noise in area 2, and includes partial edges in area 3. In order to obtain suitable edge features for desired edge outputs, a method to extract good edge features efficiently and effectively is required.

2.3 Related Work on GP for Edge Feature Construction

The existing limited work in edge detection using GP mainly focuses on construction of low-level edge features. Also, GP has been used to construct composite edge features.



(a)69020.jpg (b)T_{gg} (c)T_{hg} (d)T_{sd}

Fig. 2 Detected image based on Gaussian gradients T_{gg}, histogram gradients T_{hg} and normalised standard deviations T_{sd}.

2.3.1 Low-level Edge Feature Construction

In the construction of low-level edge features, raw pixel values are combined together by a general formula to indicate whether pixels belong to edge points or non-edge points.

For example, GP has been used to approximate a target edge detector based on the responses of the target edge detector, such as the Sobel edge detector [20, 18] and the Canny edge detector [8]. Also, GP evolved specific programs to approximate desired outputs from a set of different one-dimensional step edge signals and noise [19]. These evolved programs were used to design one-dimensional filters to extract edge features. Like regression problems, GP is used for approximation of desired outputs, such as outputs of existing detectors or desired responses on signals. However, the desired outputs are obtained based on the understanding of edge responses. Only limited edge information can be used to construct edge features when GP is employed to approximate desired edge responses.

GP has also been used to extract edge features based on detection accuracy. Edge detection is considered as a special binary classification. Each pixel is discriminated as an edge point or a non-edge point. Similar to the four macros for searching a pixel's neighbours suggested by Poli [31], our previous work has employed search operators to select pixels to construct edge detectors [10, 12]. Pixels in a 13×13 window were selected to extract edge features by GP using ground truth [38]. 64-bit digital circuits as programs were evolved by GP from artificial images [16]. In [36], morphological operators erosion and dilation were used as the terminal set and the detectors evolved by GP classify pixels as edge points or non-edge points. When GP is used to extract edge features based on ground truth, the prior domain knowledge on edges is weak. However, low-level edge features usually need to be further improved.

2.3.2 Composite Edge Feature Construction

To detect object boundaries, some image operators were utilised to combine high-level detectors by GP [22]. A rotation-variant feature evolved by GP in [22] was used to train a logistic regression classifier with texture gradients. However, only one composite feature (combined with texture gradients) in [22] was presented to compete with other edge and contour detectors. The variant feature is based on multiple directions, which leads to high computational cost. In addition, the GP feature needs to be combined with other existing approaches to perform boundary detection. In our previous work [11], three rotation-invariant basic features have been constructed into composite features. The composite features constructed by GP have the advantages from the basic features, and avoided some disadvantages from them. In [11], the composite features from GP are better than the combination of a Bayesian model. How to effectively combine basic features still needs to be investigated. In additional, GP was used to design programs which start at a pixel and then "walk" to find edge points [4].

3 The Approach

In order to employ existing techniques to combine basic features into composite features, functions or terminals based on existing techniques are designed in the proposed GP system. The function set includes functions implementing a Bayesian model and general algebraic operators. The terminal set includes not only basic features, but also a simple Bayesian model.

3.1 Bayesian Function and Terminal

Edge detection is considered here as a binary classification problem. Let $j = 0$ be the class "non-edge point", $j = 1$ be the class "edge point", and x be a d-component vector-valued random variable. P_j is the prior probability for the class non-edge point or edge point. Let $p_{x|j}$ be the state-conditional probability density function for x (given $j = 0$ or 1 being the true state). The posterior probability $p_{j|x}$ can be estimated from $p_{x|j}$ by Bayesian inversion (Equation (8)) [7]. Here the conditional probability density function $p_{x|j}$ is estimated by the general multivariate normal density (see Equation (9)), where, $\hat{\mu}_j$ is the d-component sample mean vector for class j, $\hat{\Sigma}_j$ is the d-by-d sample covariance matrix, $|\hat{\Sigma}_j|$ is its

determinant and $\hat{\Sigma}_j^{-1}$ is its inverse, and $(x-\hat{\mu}_j)^T$ is the transpose of $x-\hat{\mu}_j$. Note that $|\hat{\Sigma}_j|$ might be 0. To avoid this situation, each diagonal element in $\hat{\Sigma}_j$ has a very small tolerance value $\epsilon = 1.0e-12$ added to it.

$$p_{j|x} = \frac{p_{x|j}P_j}{\sum_{k=0}^{1} p_{x|k}P_k} \tag{8}$$

$$p_{x|j} = \frac{1}{(2\pi)^{\frac{d}{2}}|\hat{\Sigma}_j|^{\frac{1}{2}}} \exp\left(-\tfrac{1}{2}(x-\hat{\mu}_j)^T\hat{\Sigma}_j^{-1}(x-\hat{\mu}_j)\right) \tag{9}$$

In order to improve the performance of the standard Bayesian model, a Bayesian function is defined in Equation (10). Here, s is the function argument, and x is one of the possible combinations from the basic features T_{gg}, T_{sd}, and T_{hg}. The function argument s is one basic feature or a subtree. Since the class edge point is the major class in edge detection, the output is a one-dimensional variable based on the posterior probability of the class edge point. In edge detection, multiple outputs are not necessary, but the function can support multiple outputs, which is a potential future application to multiple classification problems.

$$b_{1|x}(s) = \frac{p_{1|x,s}}{\sum_{j=0}^{1} p_{j|x,s}} \tag{10}$$

The range of the output for the function $b_{1|x}(s)$ is from 0 to 1, and a composite feature from the simple Bayesian model [7] can be described as $b_{1|x_1}(T_{hg})$, where $x_1 = \{T_{gg}, T_{sd}\}$. Note that the argument s of the Bayesian function $b_{1|x}$ can be the output of another $b_{1|x}$, such as $b_{1|x}(b_{1|x}(s))$.

Therefore, the Bayesian function has three behaviours: (1) selecting basic features; (2) combining basic features and s (possibly as an intermediate combination); and (3) giving edge responses after estimating the relevant multivariate normal density.

In order to check whether the simple Bayesian model as a terminal is good to combine basic edge features, $b_{1|x_1}(T_{hg})$ is also considered as a terminal. Here x_{bayes} indicates $b_{1|x_1}(T_{hg})$ used as a terminal. Since x_{bayes} combines different basic edge features, the GP system allows a tree only consisting of a single terminal, e.g., x_{bayes} (the simple Bayesian model) as a complete program.

3.1.1 General Algebraic Operators

In order to enrich the evolved Bayesian programs, ordinary algebraic operators need to be added into the function set for constructing composite features. However, the constructed composite features are expected to represent soft edge maps. A sparse and large range of the values of a constructed feature is not suitable to represent soft edge maps. Our previous work [13] has addressed this problem. Also, it is found that directly using observations of evolved programs with normal algebraic operators was not good to construct soft edge maps [11]. The transformation by a function used in the Bayesian GP system requires the mapped values to be located in a suitable range. In general, a feature, which is used to represent soft edge maps, is normalised from 0 to 1. In the simple Bayesian model, the output of a Bayesian function or x_{bayes} is from 0 to 1. All operations in the Bayesian GP system are required to map their inputs into the space from 0 to 1.

For linear operations, two general algebraic operators $\{\oplus, \ominus\}$ are added into the function set. The operation of $\oplus$ is defined in Equation (11), and the operation of $\ominus$ is defined in Equation (12). Here, s_a and s_b are the arguments of the operators $\oplus$ and $\ominus$, and the operator $\ominus$ is the absolute value of difference of s_a and s_b. The ranges of s_a and s_b are from 0 to 1, so the range of the outputs of both operators is also from 0 to 1.

$$s_a \oplus s_b = \frac{s_a + s_b}{2} \tag{11}$$

$$s_a \ominus s_b = |s_a - s_b| \tag{12}$$

Non-linear general algebraic operators could be used in the function set. Since this paper only focuses on the Bayesian model, the application of using different general algebraic operators will be investigated in the future work.

3.2 The Other Terminals

The basic features are considered as terminals and they are normalised from 0 to 1. Note that random constants are not included in the terminal set because a Bayesian program estimates the relevant multivariate density to construct a composite feature, rather than directly using its observations as a feature. Also, the outputs for all operations in the GP system are located in the range from 0 to 1. The operator $\oplus$ will automatically adjust the scales for its input (such as Equation (11)) so that its output is still located in the range from 0 to 1.

3.3 Fitness Function

From the set of terminals and functions, the range of outputs of a program is from 0 to 1, which is directly considered as the value of a composite feature. The aim of the new constructed features is to detect as many true edge points as possible, so the fitness function Fit is defined in Equation (13). Here, recall p_{rec} is the number of pixels on the edges correctly detected as a proportion of the total number of pixels on the edges, and

specificity p_{spe} is the number of pixels not on the edges correctly detected as a proportion of the total number of pixels not on the edges. Recall p_{rec} and specificity p_{spe} are calculated when a threshold is used to discriminate outputs of a program as edge points or non-edge points.

$$Fit = \frac{2p_{rec}p_{spe}}{p_{rec} + p_{spe}} \tag{13}$$

Since there are no random constants in the terminal set and the range of observations of evolved programs is restricted from 0 to 1, multiple thresholds are used to find the maximum of Fit. Here, only three thresholds are employed to find the maximum value of Fit as the fitness for the relevant program, aiming at evolving programs with high contrast edge responses. The three thresholds are 0.25, 0.5 and 0.75. In order to check whether three thresholds are enough for evaluating evolved programs, 30 thresholds ($\frac{i}{31}, i = 1, 2, \ldots, 30$) are also used to find maximum Fit. To distinguish the two different settings on Fit, Fit_3 indicates Fit with the three thresholds, and Fit_{30} indicates Fit with the 30 thresholds.

Different from our previous work [14], the Bayesian function is considered not only a function node, but also a terminal node in the Bayesian GP system. Also, Fit is allowed to use different numbers of thresholds. To further investigate the Bayesian function, there are different settings for the function set and the terminal set in the GP system. The details of these settings are given in the next section.

4 Experiment Design

We now describe the benchmark image dataset used in this paper, also give the settings for the experiments.

4.1 Image Datasets

A natural image dataset, namely the Berkeley Segmentation Dataset (BSD) [28], is employed for edge detection. Each image in the BSD dataset has 481×321 pixels and its own ground truth from different human observations. The ground truth are combined from five to ten persons as graylevel images for fairness of judgement of edges. This dataset has been popularly used for boundary detection [24] and image segmentation [1]. The training dataset and the test dataset are separated. There are 200 images in the training dataset and 100 different images in the test dataset. Fig. 3 shows six images from the training dataset and their ground truth. Note that the widths of some true edges are more than one pixel because of the combinations of several human observations. The pixels with graylevel 0 (dark) in the ground truth are non-edge points, and the others are edge points. Pixels from the training images are sampled as training dataset, which is the same as in [11]. The settings of the three basic features are the same as in [11] as well.

The true edges in the BSD dataset come from human segmentation results, they are more closer for boundary detection, and not exact for edge detection. However, boundaries are usually extracted based on the edges detected by an edge detector [30, 27]. The edges detected by an edge detection algorithm are affected the performance of detected boundaries. One way in [34] has been proposed to evaluate edge detection through boundary detection. Also, following in the suggestion of directly evaluating edge features in [29], we directly use the ground truth as the desired outputs to do the performance evaluation on extracted edge features.

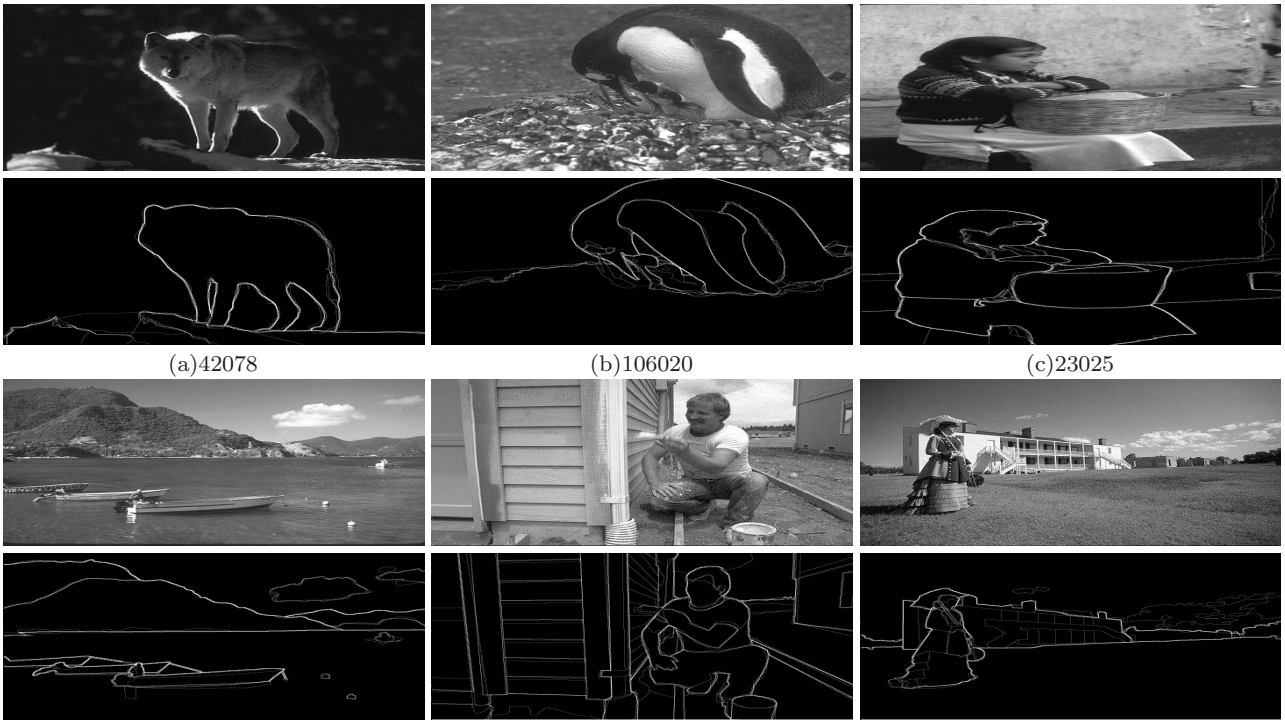
Table 1 Settings for terminals and functions in the Bayesian GP system.

Setting		Terminals	Functions
Bayesian Function	$Set_{bf,rand}$	$\{x\}$	$\{b_{1\|x}, \oplus, \ominus\}$
	$Set_{bf,all}$	$\{x\}$	$\{b_{1\|x_{all}}, \oplus, \ominus\}$
Bayesian Terminal	Set_{bt}	$\{x, x_{bayes}\}$	$\{\oplus, \ominus\}$
	$Set_{bt,all}$	$\{x, x_{bayes}\}$	$\{b_{1\|x_{all}}, \oplus, \ominus\}$
	$Set_{bt,rand}$	$\{x, x_{bayes}\}$	$\{b_{1\|x}, \oplus, \ominus\}$
Full Set	Set_{full}	$\{x, x_{bayes}\}$	$\{b_{1\|x}, b_{1\|x_{all}}, \oplus, \ominus\}$

4.2 Experiment Settings

Here, x_{all} indicates that all basic features (T_{gg}, T_{sd} and T_{hg}) are used in the terminal set. Also, $b_{1|x_{all}}$ indicates that the Bayesian function always includes all basic features, and $b_{1|x}$ indicates that the Bayesian function randomly selects one of all combinations of the basic features. In order to investigate the influence of different terminals and functions, different settings for the terminal and function sets are listed in Table 1.

Firstly, the Bayesian functions are investigated. Setting $Set_{bf,rand}$ automatically selects basic features to construct composite features. Setting $Set_{bf,all}$ restricts that all basic features must be included in a Bayesian function. The purpose for different settings on Bayesian functions is to find the difference between a large space (using $b_{1|x}$ to include possible combinations of basic features) and a narrow space (must use all basic features) for feature construction. Secondly, three settings Set_{bt}, $Set_{bt,all}$ and $Set_{bt,rand}$ are used to investigate the relationship between the terminal x_{bayes} and the functions



(a)42078 (b)106020 (c)23025

(d)68077 (e) 23080 (f)216053

Fig. 3 Six example training images from the BSD dataset and their ground truth.

$b_{1|x}$ and $b_{1|x_{all}}$. Lastly, a full set of terminals and functions is given in setting Set_{full}.

The parameter values for GP are: population size 50; maximum generations 50; maximum depth (of a program) 5, probabilities for mutation 0.15, crossover 0.80 and elitism (reproduction) 0.05. These values are chosen based on common settings and initial experiments [10]. Note that a Bayesian program has some ability to combine the basic features after estimating sample means and standard distributions in the program. A larger population and a larger number of generations are not necessarily required.

Each experiment is repeated 30 independent runs. Here, the test performance employs the F-measure technique [28]. The F-measure technique combines recall p_{rec} and precision p_{pre}, and it is defined in Equation (14). Here, precision p_{pre} is the number of pixels on the edges correctly detected as a proportion of the total number of pixels detected as edge points, and α is a factor from 0 to 1. α is usually set to 0.5 [28, 6], which is also used in this paper. Based on thresholds $\frac{k}{52}, k = 1, 2, \ldots, 51$, the maximum F_{max} of F values is considered as the test performance of a constructed feature.

$$F = \frac{p_{rec}p_{pre}}{\alpha p_{rec} + (1-\alpha)p_{pre}} \quad (14)$$

5 Results

This section provides experiment results with discussions. The results from Fit_3 will be given, then the results are compared with existing techniques. A comparison between Fit_3 (three thresholds) and Fit_{30} (30 thresholds) will be conducted. After showing detected images from Fit_3 and Fit_{30}, an example evolved Bayesian program will be interpreted.

5.1 Overall Results From Fit_3

Table 2 gives the means, standard deviations (s.d.), the maximum ("Max"), and the minimum ("Min") of F_{max} values of the evolved programs for the six settings when fitness function Fit_3 is used. The "t-test" column reports the p-values obtained from the comparisons between the relevant results (first column as the first group) and the results from $Set_{bf,rand}$ (as the second group) by using two-sample t-tests; and "MWW" reports the p-values obtained from the comparisons between the relevant results and the results from $Set_{bf,rand}$ by using Mann-Whitney-Wilcoxon tests [25].

From an overall view, setting $Set_{bf,rand}$ has the highest mean and the maximum F_{max}. It seems that setting $Set_{bf,rand}$ has the best test performance on the BSD test images. In regards of the p-values from the

Table 2 Test performance F_{max} for the six settings using fitness function Fit_3 on the BSD test image dataset.

Setting	Mean ± s.d.	Max	Min	t-test	MWW
$Set_{bf,rand}$	0.5591 ± 0.0150	0.5847	0.5199		
$Set_{bf,all}$	0.5423 ± 0.0053	0.5504	0.5349	0.0000 ↓	0.0000 ↓
Set_{bt}	0.5391 ± 0.0000	0.5391	0.5391	0.0000 ↓	0.0000 ↓
$Set_{bt,all}$	0.5381 ± 0.0137	0.5664	0.5102	0.0000 ↓	0.0000 ↓
$Set_{bt,rand}$	0.5513 ± 0.0107	0.5733	0.5277	0.0278 ↓	0.0543
Set_{full}	0.5513 ± 0.0181	0.5841	0.5102	0.0798	0.0603

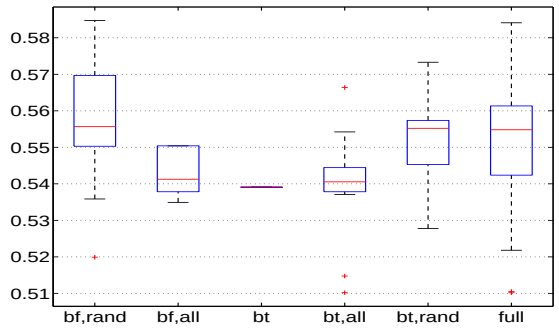


Fig. 4 Boxplots for the F_{max} values of the results (30 replications) from the six settings using fitness function Fit_3 on the 100 BSD test images.

t-tests, the results from $Set_{bf,rand}$ are significantly better than the results from the other settings, except for Set_{full}. From the Mann-Whitney-Wilcoxon tests, the results from $Set_{bf,rand}$ are also significantly better than the results from $Set_{bf,all}$, Set_{bt} and $Set_{bt,all}$. Since only settings $Set_{bf,rand}$, Set_{full} and $Set_{bt,rand}$ include $b_{1|x}$, it seems that $b_{1|x}$ is important to construct high-level features for performance improvement when fitness function Fit_3 is used.

Fig. 4 reveals the F_{max} values of the results from the six settings using Fit_3. Here the box labels are the indices of the six settings, and each box represents the relevant setting. From these boxplots, the results from Set_{bt} are the same. From Fig. 4 and Table 2, a few evolved programs from settings $Set_{bf,rand}$, $Set_{bt,all}$ and Set_{full} have very bad test performance, and their F_{max} values are less than 0.52. From an overall view, all settings, expect for Set_{bt}, have at least half of their evolved programs with F_{max} values larger than 0.54. From our previous work in [11], the highest F_{max} among the three basic features is 0.5434, and a simple Bayesian method has $F_{max} = 0.5302$, which is obviously lower than most of the evolved programs from the six settings here, except for Set_{bt}. Therefore, GP can effectively evolve Bayesian programs to improve detection performance when fitness function Fit only uses the three thresholds to find the maximum value.

5.1.1 Bayesian Function

From the comparison between $Set_{bf,rand}$ (including $b_{1|x}$) and $Set_{bf,all}$ (including $b_{1|x_{all}}$), the latter is significantly worse. If using Fit_3, the Bayesian program including all three basic features is not good to find good Bayesian programs. From the boxplots in Fig. 4, the test performances of the evolved programs from $Set_{bf,all}$ are located in a narrow range. However, the test performance of the evolved programs from $Set_{bf,rand}$ spreads over a larger range. A reason is that function $b_{1|x_{all}}$ always includes the three basic features. After estimating an evolved program including $b_{1|x_{all}}$, the test performance is strongly affected by the combination of three basic features (as a Bayesian model). Note that fitness function Fit_3 only uses three thresholds to find the maximum value, and a Bayesian model with three basic features has high contrast responses for edge points. However, function $b_{1|x}$ randomly selects basic features, it is possible that only one basic feature is selected and the evolved program has high test performance. Since the combination of three basic features using $b_{1|x}$ are varied and the evolved programs from using $b_{1|x_{all}}$ always include all basic features, an evolved program from using function $b_{1|x}$ may have lower F_{max} than the evolved programs from using function $b_{1|x_{all}}$. From the minimum value of test performance F_{max} in Table 2 and Fig. 4, a bad evolved program exists in setting $Set_{bf,rand}$, but there are no outliers in setting $Set_{bf,all}$.

Therefore, when Fit only uses the three thresholds to find the maximum value as the fitness of an evolved program, the evolved program with $b_{1|x_{all}}$ is strongly connected to the combination of all basic features and the test performance is located in a stable range; but evolved programs with $b_{1|x}$ have flexible structures to combine basic features, which brings good test performance on some evolved programs in most cases, at the same time, leads to bad test performance on a few evolved programs.

5.1.2 Bayesian Terminal

When x_{bayes} is added into the terminal set, all evolved programs from Set_{bt} using Fit_3 are equal to the single

Table 3 Comparison between the results from $Set_{bf,rand}$ using Fit_3, image Gaussian gradients T_{gg}, normalised standard deviations T_{sd}, histogram gradients T_{hg}, Sobel edge detector and the simple Bayesian terminal x_{bayes} on the 100 BSD test images.

	F_{max}	p-value
$Set_{bf,rand}$	0.5591 ± 0.0150	
T_{gg}	0.5153	0.000 $\downarrow$
T_{sd}	0.4968	0.000 $\downarrow$
T_{hg}	0.5434	0.000 $\downarrow$
Sobel	0.4832	0.000 $\downarrow$
x_{bayes}	0.5391	0.000 $\downarrow$

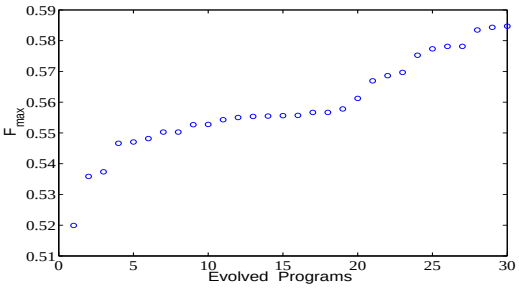


Fig. 5 F_{max} values for the features constructed by the evolved 30 programs from GP using $Set_{bf,rand}$ and Fit_3.

node tree x_{bayes}. It seems that Fit_3 could not find any better combination from x_{bayes} and the three basic features without using Bayesian functions, compared with x_{bayes}. Note that the test performance F_{max} on x_{bayes} is a bit higher than the simple Bayesian model in [11]. The change is caused by x_{bayes} adding tolerance ϵ in $\hat{\Sigma}_j$.

After adding Bayesian functions, the results from $Set_{bt,all}$ are similar to the results from $Set_{bf,all}$, and the results from $Set_{bt,rand}$ are similar to the results from $Set_{bf,rand}$. It is surprising that there are two evolved programs from $Set_{bf,all}$ with F_{max} values less than 0.52. In $Set_{bf,all}$, $b_{1|x_{all}}$ and x_{bayes} include the three basic features, but there are still two programs with bad test performance. From the boxplots in Fig. 4, the terminal x_{bayes} might be not good for constructing high-level features when fitness function Fit_3 is used. This suggests that directly combining the simple Bayesian model with all basic features might be not good for performance improvement.

5.1.3 Combination

When all functions and terminals are added into the GP system, namely using Set_{full}, the results from Set_{full} are not significantly different from the results from setting $Set_{bf,rand}$. It seems that adding x_{bayes} and $b_{1|x_{all}}$ in setting $Set_{bf,rand}$ does not affect the evolved programs when Fit_3 is used. Although the search space for candidate solutions from Set_{full} is larger than the space from $Set_{bf,rand}$, GP can still effectively discover good programs to construct high-level features.

5.2 $Set_{bf,rand}$ using Fit_3 vs Existing Techniques

Table 3 presents the comparisons between the results from $Set_{bf,rand}$ using Fit_3 and the results from T_{gg}, T_{sd}, T_{hg}, the Sobel edge detector, and x_{bayes}. The p-values are obtained from the comparison between the relevant results (first row as the first group) and the results from $Set_{bf,rand}$ using Fit_3 (as the second group) when one-sample t-tests are used. In additional, the 95% confidence interval (based on the t-test) for the evolved programs from $Set_{bf,rand}$ is $(0.5533, 0.5648)$. These results show that the features constructed by GP significantly improve the detection performance. However, the F_{max} value from x_{bayes} is lower than the basic feature T_{hg}, so it seems that the simple combination does not improve the detection performance (F_{max}). Therefore, GP is effective for automatic construction of new features, and improves the performance of using the Bayesian model to combine basic edge features for edge detection.

In order to check the details of the performance on all evolved programs, Fig. 5 gives the relevant F_{max} values for all 30 evolved Bayesian programs from $Set_{bf,rand}$ with Fit_3. Most of the evolved Bayesian programs have higher F_{max} than the best basic feature T_{hg}, and most of the evolved programs obtain good performance. However, the constructed features from three evolved programs are worse than T_{hg} (with F_{max}). The lowest F_{max} value of the constructed feature from the worst evolved program is only 0.52. How to improve the performance on the worst evolved program will be a future work.

5.2.1 Recall and Precision of an Evolved Program

Fig. 6 shows different values of recall and precision for T_{gg}, T_{sd}, T_{hg}, x_{bayes}, and a program evolved by GP (with $F_{max} = 0.5847$). Here, "@" is the position for the F_{max}. From the different thresholds, it is clear that the evolved program has higher precision than the others when recall is not too low. For T_{gg}, recall is high but precision is too low, and precision is high but recall is too low.

One interesting observation from Fig. 6 is that x_{bayes} has a narrow range for recall and precision. A reason for this phenomenon is that the Bayesian Equation (8) gives very low probabilities for most non-edge points and very high probabilities for most edge points. Therefore, the change of the threshold does not strongly af-

Table 4 Test performance F_{max} for the six settings using Fit_{30} on the BSD test image dataset.

Setting	Mean ± s.d.	Max	Min	t-test	MWW	Fit_3
$Set_{bf,rand}$	0.5754 ± 0.0134	0.5968	0.5509			0.0000 ↑
$Set_{bf,all}$	0.5707 ± 0.0104	0.5927	0.5531	0.1397	0.0726	0.0000 ↑
Set_{bt}	0.5706 ± 0.0198	0.5966	0.5087	0.2824	0.2697	0.0000 ↑
$Set_{bt,all}$	0.5706 ± 0.0166	0.5950	0.5250	0.2278	0.4705	0.0000 ↑
$Set_{bt,rand}$	0.5656 ± 0.0230	0.5946	0.4998	0.0518	0.1290	0.0053 ↑
Set_{full}	0.5684 ± 0.0159	0.5939	0.5087	0.0743	0.2697	0.0006 ↑

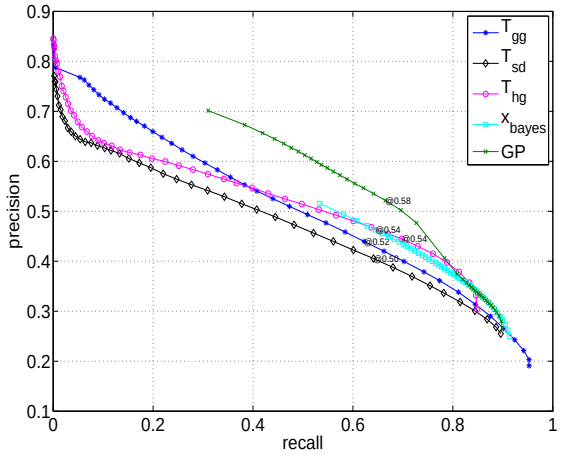


Fig. 6 Recall and precision for T_{gg}, T_{sd}, T_{hg}, x_{bayes}, and an evolved program.

fect the detection performance. Compared with the basic features, x_{bayes} and the evolved program have very high recall values for many thresholds. It seems that the composite features from the Bayesian model can easily discriminate pixels as edge points or non-edge points with a set of thresholds. This is similar to the characteristics of edge responses (high contrast) on the composite features constructed by using Gaussian distribution estimation. Whether the multivariate density using other distributions can be used to obtain rich edge responses will be investigated in the future.

5.3 Fitness Functions Fit_3 vs Fit_{30}

Table 4 gives the means, standard deviations, maximum and minimum of F_{max} values for the six settings using Fit_{30}. Here, "t-test" indicates p-values obtained from the comparisons between the relevant results (setting in the first column as the first group) and the results from $Set_{bf,rand}$ using two-sample t-tests, MWW indicates p-values from their comparisons with the Mann-Whitney-Wilcoxon tests, and "Fit_3" indicates p-values obtained from the comparisons between the results from Fit_{30} and Fit_3 (using the same setting) with paired-sample t-tests. The paired-sample t-tests are used for the comparisons between Fit_{30} and Fit_3 because of the same initial population.

There are some interesting observations from Table 4. Firstly, the test results from Fit_{30} are significantly better than the results from Fit_3. It seems that the evaluation based on three thresholds is not good to find good features (in terms of F_{max}), although Bayesian functions or terminals give high contrast responses. A potential reason is that Bayesian outputs are a combination of basic features with the two general algebraic operators. Since combining the three basic features needs multiple thresholds to find the maximum Fit values, a program, including the combination of Bayesian subtrees and subtrees constructed by basic features and the two general algebraic operators, might give rich responses on edge points and non-edge points. After using 30 thresholds to find the maximum of Fit as fitness for evolved programs, subtrees constructed by basic features and the two general algebraic operators might strongly affect the fitness of the relevant program.

Secondly, when Fit_{30} is used, there are no significant differences between the results from $Set_{bf,rand}$ and the other settings. Using multiple thresholds is easier to find a threshold which is closer to the optimal threshold for Fit. Some evolved programs evaluated by Fit_3 are not good, but they have good binary outputs by another threshold, not including in the thresholds used in Fit_3. Since Fit_{30} is more reasonable than Fit_3 to evaluate programs' fitness, all settings have good results. From the t-tests and the Mann-Whitney-Wilcoxon tests, there is no influence from using the Bayesian node in the function set or the terminal set. The way of selecting the full set of basic features or randomly selecting basic features does not obviously affect the test performance of the evolved problems.

Thirdly, the maximum F_{max} in each setting is increased. It seems that the ability to find good programs to construct high-level features is improved after using Fit_{30}.

Lastly, the worst evolved programs from settings $Set_{bf,rand}$ and $Set_{bf,all}$ are improved obviously when Fit_{30} replaces Fit_3, but the worst evolved programs from settings Set_{bt}, $Set_{bt,rand}$ and Set_{full} become even

worse, in terms of F_{max}. It is interesting that multiple thresholds do not improve the test performance of the worst evolved programs for the three settings. From each setting including x_{bayes}, the worst test performance from their results is lower than the test performance on each of the basic features. Therefore, the simple Bayesian model with all basic features might be not good to use as a terminal, and it is suggested that the Bayesian combination technique should only be used as a function to combine basic features.

5.4 Detected Images

Fig. 7 shows two example detected images from the three basic features, and the best evolved programs from Fit_3 and Fit_{30} with setting $Set_{bf,rand}$. From the detected images, the detected images by x_{bayes} and the evolved programs have obviously strong responses on true edge points, which is similar to T_{hg}. The strong responses on non-edge points from T_{hg} are mostly suppressed by x_{bayes} and the evolved programs. However, some non-edge points with high responses from T_{hg} still have high responses in x_{bayes}. Comparing the responses from T_{sd} and T_{gg}, most of the non-edge points with high responses from T_{hg} in these two images are obviously decreased in the GP evolved programs. Therefore, the Bayesian GP system effectively constructs new composite features.

From the visual results detected by the program from Fit_3, the strong responses on non-edge points from discontinuous background areas (such as T_{hg}) are decreased. Comparing with visual results from Fit_3, the responses on these non-edge points are further decreased in the two images detected by the evolved program from Fit_{30}. However, the weak responses on non-edge points from the background in the two detected images from Fit_{30} are increased a bit. Since these increased responses are still obviously weaker than the responses on edge points, the influence can be neglected. Also, from the subjective view, the detected images from Fit_{30} is thinner than the detected images from Fit_3. These observations suggest that the multiple thresholds in Fit can improve the ability of finding better programs to construct features.

Fig. 8 shows the image 69020 detected by the best evolved program from $Set_{bf,rand}$ with Fit_3 and the combination x_{bayes}. As can be seen, the detected image by the best evolved program from GP is improved in areas 1 and 2. Also, the responses on edge points are obviously higher than the non-edge points. Most of the true non-edge points have very weak responses (very dark) for this image, which is the same as the detection result from x_{bayes}. Although x_{bayes} decreases responses on most of the non-edge points, it wrongly strengthens responses on a few non-edge points.

5.5 Example Evolved Bayesian Program

The best evolved Bayesian GP program from $Set_{bf,rand}$ with Fit_3 is described by Equation (15), where BG_{bayes_1} indicates an evolved Bayesian program, and subscript indices 0, 1 and 2 represent the basic features T_{gg}, T_{sd} and T_{hg} (in b_x, abbreviated from $b_{1|x}$), respectively. Function $b_{0,1}(T_{gg})$ is used to construct a model with the multivariate including T_{gg} and T_{sd}. Note that the proposed function accepts redundant variables, such as $x = \{T_{gg}, T_{gg}\}$. From the first part $b_{0,1,2}(\ b_{0,1}(b_{0,1}(T_{gg})))$, the formula aims at finding true edge points detected by T_{gg} and then constructing a feature with three basic variables. Since T_{gg} has high recall, this first part should be good at finding true edge points. The second part $b_2(b_2(b_2(T_{hg})))$ only includes the basic feature T_{hg}. After repeating estimation on true edge points (using the second part), the responses for those non-edge points with high responses in T_{hg} should be decreased. Since the output of the Bayesian function is based on the class edge point, the accuracy (precision) for the constructed feature should not be too low. Generally, the responses on clear boundaries (easily detected) are accurately given. Therefore, the second part possibly focuses on the boundary detection.

$$BG_{bayes_1} = b_{0,1,2}(b_{0,1}(b_{0,1}(T_{gg})))\oplus b_2(b_2(b_2(T_{hg}))) \quad (15)$$

In order to visually present the characteristics from the two parts, Fig. 9 shows two example images detected by the two parts of the best evolved method. "First" means the images detected by the first part $b_{0,1,2}(\ b_{0,1}(b_{0,1}(T_{gg})))$, and "Second" for the second part $b_2(b_2(b_2(T_{hg})))$. From the two example detected images, the second part appears to be better than the first part to detect true edge points. Since the first part focuses on finding edge points (possibly with low precision), the combination of both parts gives high responses on boundaries and low responses on non-edge points. The test performances (F_{max}) on the first part and second part for the 100 BSD test images are 0.5231 and 0.5436, and the complete BG_{bayes_1} has $F_{max} = 0.5847$. Compared with T_{hg}, the second part does not improve the F_{max} value, but it decreases response magnitudes for non-edge points. Iteratively estimating a single feature might be helpful to improve detection performance, which is a future work.

In order to further check the performance of both parts, Fig. 10 reveals the details of recall and precision for the two parts of the evolved programs. From the

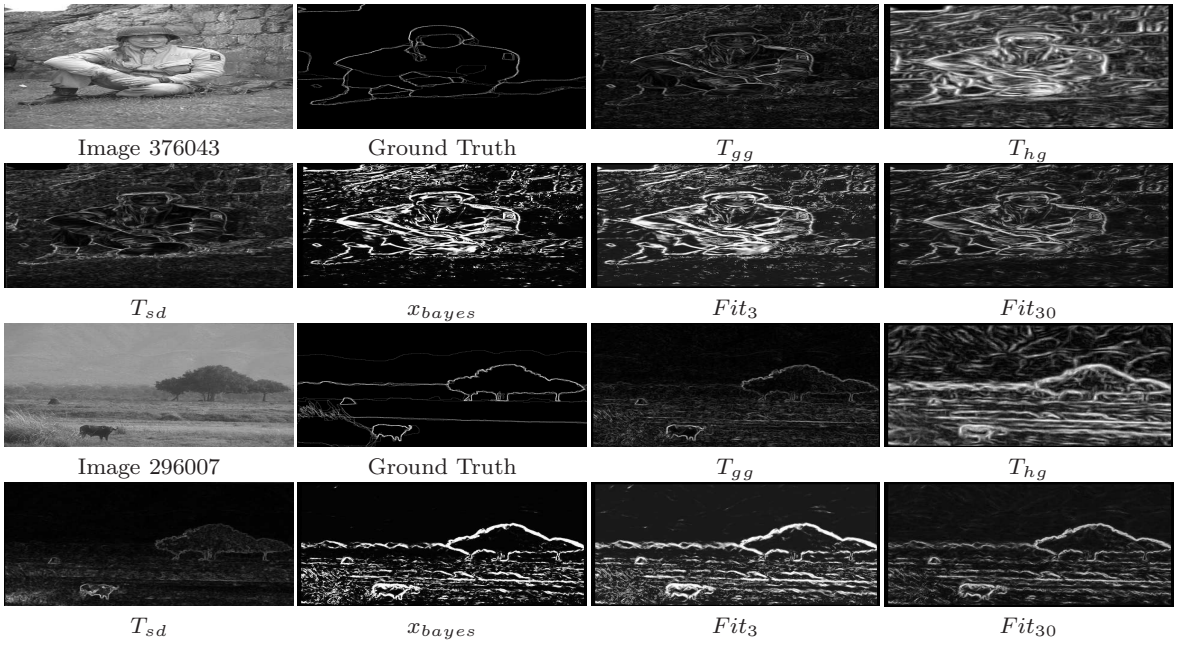
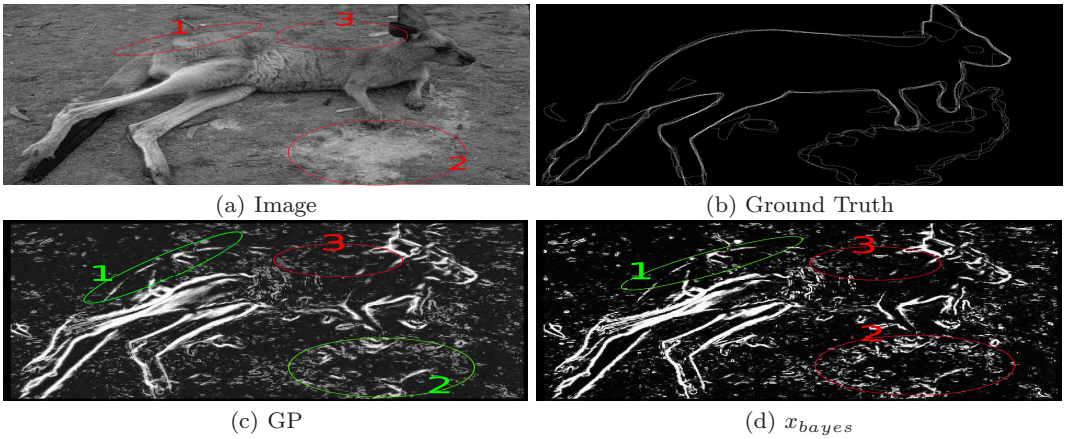


Fig. 7 Two example images detected by T_{gg}, T_{hg}, T_{sd}, x_{bayes}, and the best evolved programs from GP with $Set_{bf,rand}$ using Fit_3 and Fit_{30}.



(a) Image (b) Ground Truth

(c) GP (d) x_{bayes}

Fig. 8 Image 69020 detected by an evolved program from $Set_{bf,rand}$ with Fit_3 and the combination x_{bayes}.

curves, it is found that the recall and precision values of the second part are almost located on the curve of T_{hg}. Different from T_{hg}, the recall and precision values of the second part of the evolved program crowd in an area (with not too low recall and not too low precision). From the view of the performance based on different thresholds, repeating estimation of a single variable with its estimated outputs transforms the variable values into a suitable range which is not strongly affected by the change of a threshold. Although neither the first part nor the second part is better than T_{hg}, the program combining the two parts is clearly significantly better.

6 Further Discussions

This section discusses the influence of the general algebraic operators, computational cost for the settings, and convergence of the GP system.

6.1 Influence of General Algebraic Operators

The results from Fit_3 and Fit_{30} (in Tables 2 and 4) show that multiple thresholds obviously affect the test performance of the evolved programs. When Set_{bt} and Fit_3 are used, the GP system only finds x_{bayes} as the final solution. However, when Set_{bt} and Fit_{30} are used, the evolved programs are significantly better the results from Set_{bt} with Fit_3.

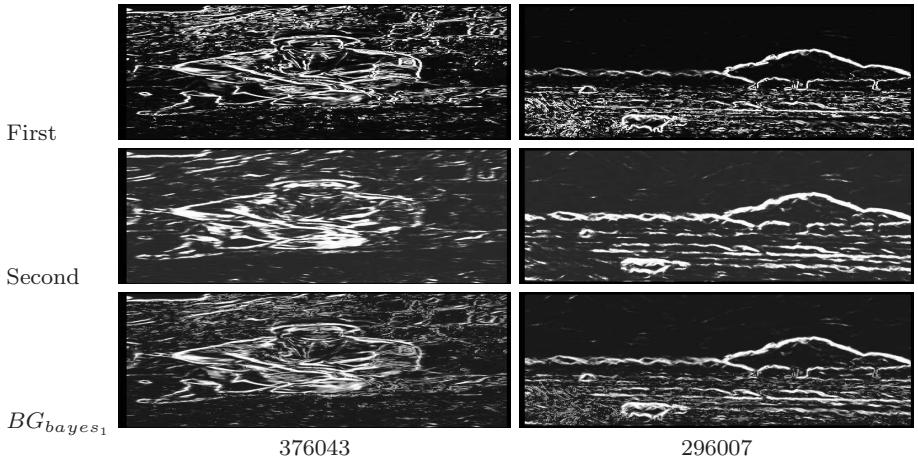


Fig. 9 Two example images detected by two parts of the best evolved method from GP.

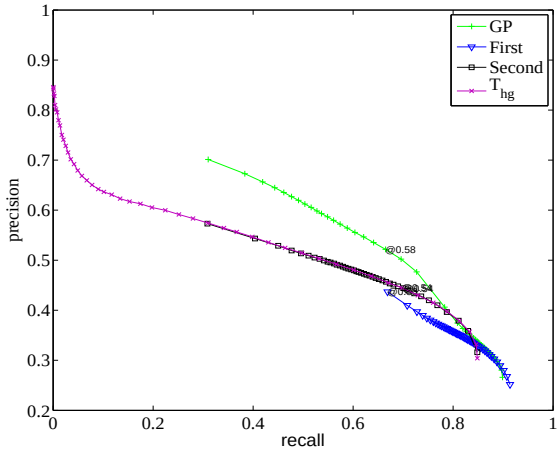


Fig. 10 Recall and precision for an evolved program, its two parts and T_{hg}.

Since Fit_3 mainly focuses on selecting programs with high contrast edge responses, the two general algebraic operators do not work well on combinations of basic features. If the terminal set only includes x and the function set only includes the two general algebraic operators, the test performance (F_{max}) for using Fit_3 is 0.5459 ± 0.0087. From the evolved programs, it is found that 25 of the 30 evolved programs are equal to T_{hg}. Therefore, it is hard to find a good combination of basic features when Fit_3 is used. This suggests that combining basic features with the two general algebraic operators might not be good to construct composite features with high contrast edge responses.

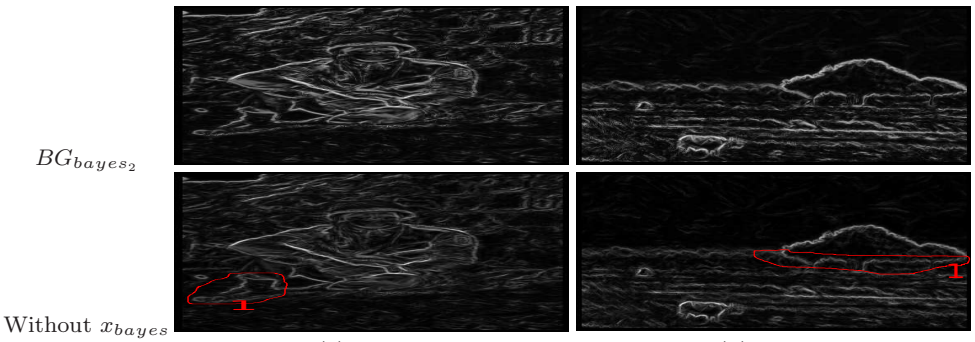
When Fit_{30} is used to evolve programs, the terminal set only includes x, and the function set only includes the two general algebraic operators, the test performance (F_{max}) is 0.5714 ± 0.0125. As can be seen, only using the two operators can also obtain good composite features. The test performance on Set_{bt} with Fit_{30} is very close to the test performance without using x_{bayes}. In order to check whether the evolved programs from Set_{bt} with Fit_{30} include any x_{bayes}, the evolved programs are simplified. From the simplification, the number of the evolved programs without x_{bayes} is only nine out of 30 (less than one third). Therefore, x_{bayes} is still helpful to construct good composite features. Equation (16) gives an evolved program BG_{bayes_2} ($F_{max} = 0.5941$) from Set_{bt} with Fit_{30}, where $\frac{3}{8}$ and $\frac{5}{8}$ are scale parameters after simplification. The evolved program BG_{bayes_2} includes x_{bayes}, T_{hg} and T_{gg}.

$$BG_{bayes_2} = \frac{3T_{gg}}{8} \oplus \frac{5T_{hg}}{8} \oplus [(T_{hg} \ominus T_{gg}) \ominus (x_{bayes} \oplus T_{hg})] \quad (16)$$

Fig. 11 shows two example images detected by BG_{bayes_2}, and an evolved program without x_{bayes} (from Set_{bt} using Fit_{30}, and $F_{max} = 0.5940$). Comparing the visual results, the red circled areas (marked as number 1) in the two images detected by the evolved program without x_{bayes} have lower edge responses than the relevant results from BG_{bayes_2}. A reason is that the responses on composite features combined by the general algebraic operators are dependent on the responses in the basic features, but Bayesian programs increase the edge response contrast. Since the technique only using general algebraic operators is different from the Bayesian technique, further investigation on both techniques in the GP system will be conducted in the future.

6.2 Computational Cost

Table 5 gives the training time (mean ± standard deviations in seconds) for evolving programs by using the six settings respectively. Here, the p-value is obtained from



(a) 376043 (b) 296007

Fig. 11 Two example images detected by two evolved programs from Set_{bt} using Fit_{30}.

Table 5 Training time (means ± standard deviations in seconds) from Fit_3 and Fit_{30} using the six settings.

Setting	Fit_3	Fit_{30}	p-value
$Set_{bf,rand}$	190.9207 ± 61.6907	19.8340 ± 13.6920	0.0000 ↓
$Set_{bf,all}$	73.6683 ± 10.6519	14.9423 ± 12.3262	0.0000 ↓
Set_{bt}	47.2473 ± 11.3341	73.2913 ± 47.6183	0.0131 ↑
$Set_{bt,all}$	314.3730 ± 74.1631	53.6197 ± 46.0516	0.0000 ↓
$Set_{bt,rand}$	111.5797 ± 16.8419	63.4163 ± 41.6157	0.0000 ↓
Set_{full}	167.5110 ± 38.1950	43.1970 ± 38.7859	0.0000 ↓

the comparison between the results from Fit_3 and Fit_{30} in each setting by using a paired-sample t-test because of the same initial population. Note that ↓ means that the training time of using Fit_3 is significantly longer than the training time of using Fit_{30}, and ↑ means that the training from the former is significantly shorter than the latter. All experiments are run on single machines with CPU 3.1 GHz at the full speed state (the initial CPU speed is 1.6 GHz for the power saving feature). Note that the value for Fit_3 or Fit_{30} is calculated after only visiting the training data once, so the calculation cost on Fit_3 and Fit_{30} is very close when evaluating the same program.

From an overall view, the training time for each setting is quite short. The test time for detecting an image can be ignored (several milliseconds) because all basic features are loaded into the memory before executing a program.

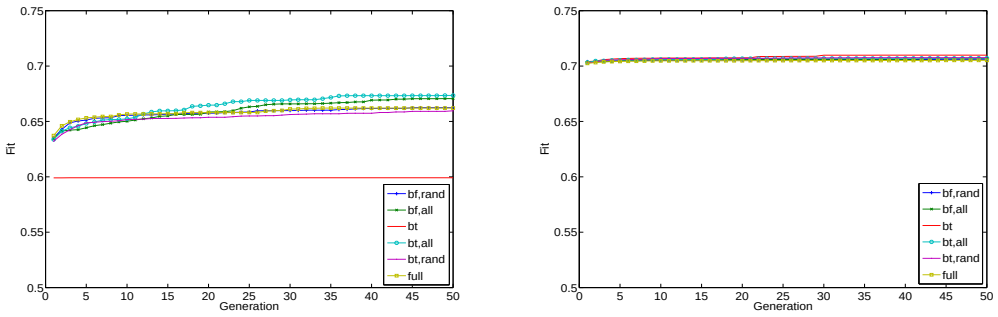
From the table, the training time of using Fit_3 is significantly longer than the training time of using Fit_{30} in each setting, except for Set_{bt}. Since subtrees including the two general algebraic operators might be important to construct a good program, the number of Bayesian functions might be reduced in the evolved good programs from Fit_{30}. Note that an algebraic operator has less computational cost than a Bayesian function. Since a Bayesian function needs to calculate the sample means and standard deviations of its input, the training cost from Fit_3 is higher than the training cost from Fit_{30}, except for Set_{bt}. Since the evolved programs from Set_{bt} only do not include Bayesian functions, the training time of using Fit_{30} in Set_{bt} is longer than the training time of using Fit_3 in Set_{bt}. Also there is another reason for using Fit_{30} being longer than using Fit_3 for Set_{bt}, that is, Fit_3 in Set_{bt} only finds x_{bayes} as final solutions; but Fit_{30} in Set_{bt} needs to find good subtrees constructed by the two general algebraic operators, which takes some computational cost.

6.3 Convergence

Since the population size and maximum generation are very small in the settings, it is worth investigating the convergence of this Bayesian GP system. Fig. 12 reveals that the average best fitness at each generation in the six settings when Fit_3 or Fit_{30} is used. From Fig. 12 (a), since Set_{bt} only finds x_{bayes} as the final solutions, the best fitness value stays horizontal. The other settings almost reach constant values after generation 40 when Fit_3 is used. For Fit_{30} (see Fig. 12 (b)), all settings are approximately convergent to constant values after generation 30. $Set_{bf,rand}$ and $Set_{bf,all}$ increase a bit in fitness after generation 43. Therefore, Fig. 12 (a) and (b) indicate that the GP system is convergent or very closely convergent at generation 50.

6.4 Number of Thresholds in Fit

The fitness function Fit_3 has the evolved results with high contrast edge responses, and the evolved results



(a) Fit_3 (b) Fit_{30}

Fig. 12 Means of the best fitness at each generation in each setting.

from Fit_{30} decreases the responses on non-edge points which have strong responses in the results from Fit_3. In order to investigate the influence of the number of thresholds in Fit, ten thresholds are used to evolve Bayesian programs when $Set_{bf,rand}$ is used. Here, we use Fit_{10} to indicate Fit using ten thresholds. The test performance F_{max} for Fit_{10} is 0.5763 ± 0.0152. The results from Fit_{10} are significantly better the results from Fit_3 and are not significantly different from the results from Fit_{30}.

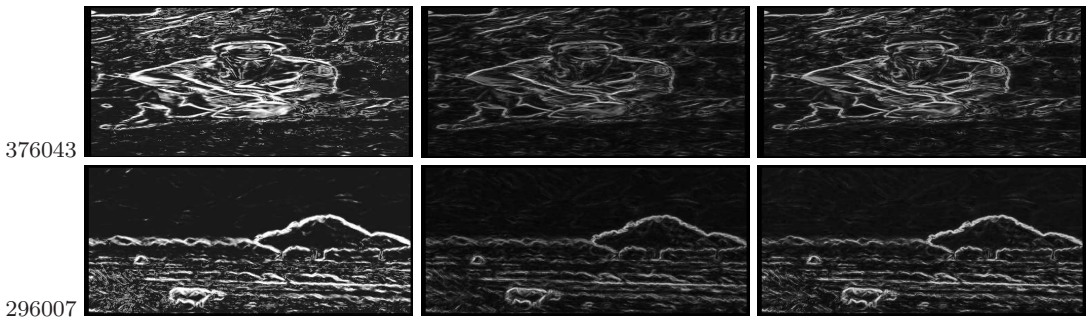
Fig. 13 shows two example detected images from Fit_3, Fit_{10} and Fit_{30}. From an overview, there is no very obvious difference between the results from Fit_{10} and Fit_{30}. Compared to the detected results from Fit_3, the detected results from Fit_{10} decrease the responses on some edges and discontinuities non-edge areas, such as the background of image 376043. Compared to the detected results from Fit_{30}, responses on some true edges in the results from Fit_{10} are lower, such as the boundary of the tree in image 296007. It is possible that using a very small number of thresholds in Fit makes a good program with the observations of most non-edge points close to 0 and the observations of most edge points close to 1.

When the number of thresholds used in Fit is neither small nor large, a good program can have the observations for edge points and non-edge points which are close but located in different ranges. Since the observations of programs are close and a range between two close thresholds is not very narrow, the edge response contrast for detected images from Fit_{10} is lower than the contrast of the detected images from Fit_3. When the number of thresholds used in Fit is large, each range between two close thresholds is narrow so that the observations of a good program for edge points and non-edge points are located in different ranges. It is possible that most distances from the observations of edge points to the observations of non-edge points in Fit_{30} are longer than the relative distances in Fit_{10}. Therefore, the edge response contrast for the detected images from Fit_{30} is higher than the relevant contrast from Fit_{10}. Note that the edge response contrast is a subjective evaluation, and it is not equal to the test performance F_{max}. To obtain very high contrast edge responses, it is suggested that a small number of thresholds should be used in Fit. However, if weak responses on discontinuous non-edge areas are also considered, it is suggested that a large number of thresholds should be used in Fit.

7 Conclusions

The goal of this paper was to investigate automatic high-level feature construction for edge detection using GP and Bayes' theorem. The Bayesian technique was employed to combine basic features via using a general multivariate normal density. The goal was successfully achieved by evolving Bayesian programs with Bayesian functions and terminals, and two general algebraic operators.

From the results, firstly, the evolved Bayesian programs are better than the simple Bayesian model directly using the multivariate normal density, and also have high contrast edge responses. Secondly, when the fitness function only uses three thresholds to search programs with high contrast edge responses, the Bayesian function with randomly selected basic features is better than the function using the full set of basic features. However, when the fitness function uses 30 thresholds, there are no significant differences in terms of the test performance F_{max}. Thirdly, in order to obtain better high contrast edge responses on composite features, using the general multivariate normal density as a function is better than using it as a terminal to evolve programs. Lastly, further analysis of the best evolved program reveals that iteratively estimating a single feature may help to improve detection performance.



(a) Fit_3 (b) Fit_{10} (c) Fit_{30}

Fig. 13 Comparisons among two detected example images from Fit_3, Fit_{10} and Fit_{30}.

For future work, we will investigate the influence of density functions used in multivariate distributions. Also, the Bayesian programs will be further manipulated for boundary detection.

References

1. Arbeláez P, Maire M, Fowlkes C, Malik J (2011) Contour detection and hierarchical image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 33(5):898–916
2. Azad P, Gockel T, Dillmann R (2008) Computer Vision: Principles and Practice. Elektor
3. Basu M (2002) Gaussian-based edge-detection methods: a survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 32(3):252–260
4. Bolis E, Zerbi C, Collet P, Louchet J, Lutton E (2001) A GP artificial ant for image processing: preliminary experiments with EASEA. In: Proceedings of the 4th European Conference on Genetic Programming, pp 246–255
5. Canny J (1986) A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 8(6):679–698
6. Dollar P, Tu Z, Belongie S (2006) Supervised learning of edges and object boundaries. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol 2, pp 1964–1971
7. Duda RO, Hart PE, Stork DG (2000) Pattern Classification (2nd Edition). Wiley-Interscience
8. Ebner M (1997) On the edge detectors for robot vision using genetic programming. In: Proceedings of Horst-Michael Groβ, Workshop SOAVE 97 - Selbstorganisation von Adaptivem Verhalten, pp 127–134
9. Fernández-García N, Carmona-Poyato A, Medina-Carnicer R, Madrid-Cuevas F (2008) Automatic generation of consensus ground truth for the comparison of edge detection techniques. Image and Vision Computing 26(4):496–511
10. Fu W, Johnston M, Zhang M (2011) Genetic programming for edge detection: a global approach. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp 254–261
11. Fu W, Johnston M, Zhang M (2012) Automatic construction of invariant features using genetic programming for edge detection. In: Proceedings of the Australasian Joint Conference on Artificial Intelligence, pp 144–155
12. Fu W, Johnston M, Zhang M (2012) Genetic programming for edge detection using blocks to extract features. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp 855–862
13. Fu W, Johnston M, Zhang M (2012) Soft edge maps from edge detectors evolved by genetic programming. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp 24–31
14. Fu W, Johnston M, Zhang M (2013) Genetic programming for edge detection using multivariate density. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp 917–924
15. Ganesan L, Bhattacharyya P (1997) Edge detection in untextured and textured images: a common computational framework. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 27(5):823–834
16. Golonek T, Grzechca D, Rutkowski J (2006) Application of genetic programming to edge detector design. In: Proceedings of the International Symposium on Circuits and Systems, pp 4683–4686
17. Grigorescu C, Petkov N, Westenberg MA (2004) Contour and boundary detection improved by surround suppression of texture edges. Image and Vision Computing 22(8):609–622
18. Harding S, Banzhaf W (2008) Genetic programming on GPUs for image processing. International Journal of High Performance Systems Architecture

1(4):231–240
19. Harris C, Buxton B (1996) Evolving edge detectors with genetic programming. In: Proceedings of the First Annual Conference on Genetic Programming, pp 309–314
20. Hollingworth G, Smith S, Tyrrell A (1999) Design of highly parallel edge detection nodes using evolutionary techniques. In: Proceedings of the Seventh Euromicro Workshop on Parallel and Distributed Processing, pp 35–42
21. Jiang L, Wang D, Cai Z, Yan X (2007) Survey of improving naive bayes for classification. In: Proceedings of the 3rd International Conference on Advanced Data Mining and Applications, pp 134–145
22. Kadar I, Ben-Shahar O, Sipper M (2009) Evolution of a local boundary detector for natural images via genetic programming and texture cues. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp 1887–1888
23. Kokare M, Biswas PK, Chatterji BN (2003) Edge based features for content based image retrieval. Pattern Recognition 36(11):2649–2661
24. Kokkinos I (2010) Boundary detection using F-measure-, filter- and feature- (F3) boost. In: Proceedings of the 11th European Conference on Computer Vision: Part II, pp 650–663
25. Kruskal WH (1957) Historical notes on the Wilcoxon unpaired two-sample test. Journal of the American Statistical Association 52(279):356–360
26. Lam L, Lee SW, Suen C (1992) Thinning methodologies-a comprehensive survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 14(9):869–885
27. Martin D, Fowlkes C, Tal D, Malik J (2001) A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proceedings of the 8th International Conference on Computer Vision, vol 2, pp 416–423
28. Martin D, Fowlkes C, Malik J (2004) Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(5):530–549
29. Moreno R, Puig D, Julia C, Garcia M (2009) A new methodology for evaluation of edge detectors. In: Proceedings of the 16th IEEE International Conference on Image Processing (ICIP), pp 2157–2160
30. Papari G, Petkov N (2011) Edge and line oriented contour detection: state of the art. Image and Vision Computing 29:79–103
31. Poli R (1996) Genetic programming for image analysis. In: Proceedings of the First Annual Conference on Genetic Programming, pp 363–368
32. Rezai-Rad G, Larijani HH (2007) A new investigation on edge detection filters operation for feature extraction under histogram equalization effect. In: Proceedings of the Geometric Modelling and Imaging, pp 149–153
33. Shyi-Chyi C, Wen-Hsiang T (1994) Image compression by moment-preserving edge detection. Pattern Recognition 27(11):1439–1449
34. Song W, Feng G, Tiecheng L (2006) Evaluating edge detection through boundary detection. EURASIP Journal on Applied Signal Processing 2006:1–15
35. Sullivan J, Carlsson S (2002) Recognizing and tracking human action. In: Proceedings of the 7th European Conference on Computer Vision-Part I, pp 629–644
36. Wang J, Tan Y (2010) A novel genetic programming based morphological image analysis algorithm. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, pp 979–980
37. Zhang M, Cagnoni S, Olague G (2009) Gecco 2009 tutorial: Evolutionary computer vision. In: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference, pp 3355–3380
38. Zhang Y, Rockett PI (2005) Evolving optimal feature extraction using multi-objective genetic programming: a methodology and preliminary study on edge detection. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp 795–802