2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Unsupervised Learning for Edge Detection using Genetic Programming

Wenlong Fu Mark Johnston Mengjie Zhang

***Abstract*— In edge detection, a machine learning algorithm generally requires training images with their ground truth or designed outputs to train an edge detector. Meanwhile the computational cost is heavy for most supervised learning algorithms in the training stage when a large set of training images is used. To learn edge detectors without ground truth and reduce the computational cost, an unsupervised Genetic Programming (GP) system is proposed for low-level edge detection. A new fitness function is developed from the energy functions in active contours. The proposed GP system utilises single images to evolve GP edge detectors, and these evolved edge detectors are used to detect edges on a large set of test images. The results of the experiments show that the proposed unsupervised learning GP system can effectively evolve good edge detectors to quickly detect edges on different natural images.**

I. Introduction

EDGE detection is an important step in image processing and computer vision [1]. It has developed over many years [2], and there are many different approaches to detecting edges [1][3]. In low-level edge detection tasks, edges are generally detected based on raw pixel values. Differentiation-based approaches have been popularly applied to edge detection [3]. These approaches are based on specific edge domain knowledge. When only weak edge domain knowledge is provided, some learning algorithms have been used for training detectors to extract edges [4][5]. In our previous work [5][6], only training images and their ground truth were provided to automatically evolve good low-level edge detectors.

When the ground truth or the desired outputs of training images are not provided, it is hard for a learning algorithm to find edge detectors. Genetic Programming (GP) as a learning algorithm has been applied to automatic construction of edge detectors since at least 1996 [7][8]. From the existing literature, the developed GP methods for edge detection are based on supervised learning. When there is no ground truth or designed outputs for training images, how to evolve edge detectors by GP needs to be investigated.

When there is no ground truth, a method to evaluate detected edges might require some degree of domain edge knowledge. To directly find edges, the active contours approach [9] uses an energy function. Active contours can find good closed edges, but there is a heavy computational cost in the active contours approach. In addition, this approach is not stable and usually only detects part of edges in an image [1][10]. Since the evaluation of edges (by an energy function) does not need ground truth, it is desirable to develop a new fitness function for GP using, in order to to quickly evaluate detected edges.

In order to reduce the computational cost of learning, one-shot learning has been applied to object recognition [11][12]. After obtaining prior knowledge from existing datasets, such as predefined feature distributions or learnt classifiers, one-shot learning can employ a minimal set of training examples to train new classifiers. Via employing prior edge knowledge, it is expected to use only a small set of images to train good GP edge detectors.

A. Goals

The goal of this paper is to investigate unsupervised learning in GP for edge detection when a low computational cost is desired in the training and test stages. To the best of our knowledge, it is the first time that unsupervised learning in GP has been investigated for edge detection. A new fitness function developed from the energy functions in active contours is proposed to select good GP edge detectors. When GP evolves edge detectors from a single image, the image gradient as the prior edge knowledge is employed. Rather than finding GP edge detectors for each image, GP evolved edge detectors from a single image will be directly used to test other images. Specifically, we would like to investigate the following research objectives.

- Whether a single training image without ground truth can be used to evolve good edge detectors by GP for directly detecting unseen images.
- Whether GP evolved edge detectors from a single image are better of finding edges on unseen images than using a single threshold chosen from a set of fixed thresholds based on the best fitness on the single image.
- Whether the GP evolved edge detectors can be reasonably interpreted by a human.

B. Organisation

In the remainder of this paper, Section II briefly describes the background of edge detection and GP. Section III introduces the proposed GP system. Section IV gives the settings of the experiments. Section V provides the results on a benchmark natural image dataset. Section VI draws conclusions and suggests future work directions.

Wenlong Fu and Mark Johnston are with the School of Mathematics, Statistics and Operation Research, Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand, (email: {Wenlong.Fu}@msor.vuw.ac.nz).

Mengjie Zhang is with the School of Engineering and Computer Science, Victoria University of Wellington.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

II. Background

This section briefly describes the background of edge detection, active contours, and related work in edge detection using GP.

A. Edge Detection

In edge detection, differentiation-based approaches have been popularly used to extract edge features [1][3]. Based on different directions, image derivatives are calculated and then they are combined as image gradients, such as in the Sobel edge detector [13] and the Canny edge detector [14].

Ganesan and Bhattacharyya [13] suggested a common computational framework using gradients to extract edges on untextured and textured images. In a 3×3 moving window, the horizontal derivative d_x is defined by Equation (1), where, a is a parameter, $\circledast$ is the convolution operator, and I is an input image. Parameter a has been used with different values, such as $a = 2$ for the Sobel detector, $a = 1$ for the Prewitt detector [15], and $a = \sqrt{2}$ for the Frei-Chen edge detector [16].

$$d_x = \begin{bmatrix} 1 & 0 & -1 \\ a & 0 & -a \\ 1 & 0 & -1 \end{bmatrix} \circledast I \tag{1}$$

It is difficult to evaluate detected edges because edge detection is a subjective task. Besides the evaluation from direct human observation, the ground truth of the detected images is a popular choice for evaluating the performance of edge detectors [4][17]. Since edge points are not independent, the smoothness and connection of edges have been used to help to extract edges, such as the Hough transform [18] and active contours [9][19]. Active contours, also called "snakes", address how to find edges (detected curves) from the influence of the curves themselves and image gradients without ground truth. In general, an energy function is defined in the active contours approach to find closed curves, and the detected curves should have minimum energy.

Let c be a set of curves in an image. An energy function $E(c)$ generally includes an internal energy $IE(c)$ (from the curves themselves) and an external energy $EE(c)$ (such as image gradients) based on the definition of active contours [9][10][19][20]. The general $E(c)$ is defined in Equation (2), where λ is a weight factor.

$$E(c) = IE(c) + \lambda EE(c) \tag{2}$$

The internal energy IE mainly focuses on the smoothness of the curves [9][10]. The image gradient is popularly used in EE. From Aubert and Kornprobst [21], $EE(c)$ is suggested to satisfy: (1) EE is regular monotonic decreasing; and (2) $EE(0) = 1$ and $\lim_{c\to\infty}(c) = 0$. The detected curves in active contours are considered as the solution of an optimisation problem (finding the minimum energy).

B. Related Work to GP for Edge Detection

GP has been employed for edge detection. However, from the existing literature, there are no reports using GP as an unsupervised learning method to evolve edge detectors. Using desired outputs or ground truth, GP has been used to evolve programs and these programs are used to detect edges on unseen images.

In order to construct low-level edge detectors, GP has been used to evolve programs based on raw pixel values. Harris and Buxton [7] designed outputs for one-dimensional step edge responses and utilised GP to evolve programs to extract edge features. GP was used to select pixels in a 13×13 moving window to construct programs to extract edge features based on multiple objectives (including detection accuracy) [22]. Using pixels in a moving window as terminals, programs were evolved by GP for detecting edges [23][24][25]. Rather than using a moving window, GP has been investigated for constructing edge detectors based on full images [6][8][26]. The evolved programs can compete with the results from the existing edge detectors, such as the Sobel edge detector [26] and the Canny edge detector [22].

Also, ordinary image operators have been employed in GP for evolving edge detectors. For instance, Gaussian filters were employed to evolve Gaussian-based edge detectors [27][28]. GP utilised morphological erosion and dilation as terminals to evolve programs for detecting edges in binary images [29][30]. The predefined edge features were used for GP to construct composite edge features [31][32].

In summary, GP has been employed to evolve edge detectors based on ground truth or desired outputs, incorporating different degrees of prior domain edge knowledge. However, it remains an open question whether GP can be used to evolve good programs to detect edges when there are no ground truth or target edge points available.

III. The Method

In order to quickly evolve an edge detector by GP, the proposed GP system is inspired by an artificial ant for image processing [25]. Our proposed GP system evolves a program based on a single given image. The evolved program is used to detect edges in other (test) images, rather than evolving new programs again based on these test images. Since the initial investigation mainly focuses on unsupervised learning, a new fitness function, modified from an energy function in active contours, is proposed to evaluate evolved programs without the ground truth of the single training image used.

A. Terminals

A GP artificial ant [25] employed actions (turn left, turn right and move) to find food. Here, edge points are also considered as food, which is similar to a artificial ant for tracking boundaries in an image. However, the boundaries found by an ant [25] include many false alarms. In order to reduce false alarms, our GP system includes different terminals to mark pixels. A program will scan pixels from left to right and from top to bottom in an image.

The first terminal in the proposed GP system is marker m based on a single pixel. While a program is marking a pixel, the pixel is marked as an edge point if marker m is called. In order to check whether a program can quickly mark a set of pixels, a new terminal, named marker mH, is used to mark n horizontally connected pixels as edge points, and their neighbours (top and bottom positions) are marked as non-edge points. Also, another new terminal, named marker mV, is used to mark n vertically connected pixels as edge points, and their neighbours (left and right positions) are marked as non-edge points.

In order to mark pixels as non-edge points, a new terminal, called marker nE, marks the current discriminated pixel as a non-edge point. Another new terminal, called marker anE, marks all pixels in a $n \times n$ window as non-edge points.

In summary, the terminal set in the proposed GP system includes $\{m, mH, mV, nE, anE\}$.

B. Functions

In order to choose different terminals, logical functions should be used in the function set. To combine conditions into logical operation IF, a function $IFC(f, t, P1, P2)$ is designed, where f is a specific feature extracted from the current discriminated pixel and its neighbours, t is a random constant (threshold), and $P1$ and $P2$ are sub-programs. In function $IFC(f, t, P1, P2)$, $P1$ is executed when $f < t$; otherwise, $P2$ is executed.

Based on the terminal set, the conditions should include information that could be helpful to discriminate the current pixels as edge points or non-edge points and the information of the edge orientation of the discriminated pixels. Therefore, the horizontal derivative d_x and the vertical derivative d_y are selected as specific features f. Similar to the Prewitt edge detector [15], the horizontal derivative d_x for a pixel at the position (x, y) in image I is defined in Equation (3), where, k is the half of the window width, namely $k = \frac{n-1}{2}$ (n is odd), and $I_{x,y}$ is the intensity of pixel (x, y). The vertical derivative d_y is defined in Equation (4).

$$d_x(x, y) = \sum_{j=-k}^{j=k} \sum_{i=1}^{i=k} (I_{x-i,y+j} - I_{x+i,y+j}) \tag{3}$$

$$d_y(x, y) = \sum_{j=1}^{j=k} \sum_{i=-k}^{i=k} (I_{x+i,y+j} - I_{x+i,y-j}) \tag{4}$$

Considering the range of threshold t (combined with different specific features f as conditions), we rescale the range of the derivatives d_x and d_y. The rescaled derivatives d_x and d_y are defined in Equations (5) and (6).

$$d_x(x, y) = \frac{\sum_{j=-k}^{j=k} \sum_{i=1}^{i=k} (I_{x-i,y+j} - I_{x+i,y+j})}{n(n-1)} \tag{5}$$

$$d_y(x, y) = \frac{\sum_{j=1}^{j=k} \sum_{i=-k}^{i=k} (I_{x+i,y+j} - I_{x+i,y-j})}{n(n-1)} \tag{6}$$

To directly discriminate pixels without edge direction information, the image gradient g is also considered as a specific feature f. The image gradient g is defined in Equation (7). Here, we also rescale the image gradient g so that the image gradient has the same range as the derivatives d_x and d_y when only the positive values are considered.

$$g = \sqrt{\frac{d_x^2 + d_y^2}{2}} \tag{7}$$

When the variance or the standard deviation of pixel intensities in a window is very low, the center pixel in the window should be a non-edge point. Therefore, the variance sd is also employed as a specific feature f. Here, sd is extracted from a $n \times n$ window, and it is defined in Equation (9), where r_{sd} is a scale factor for sd.

$$sum(x, y) = \textstyle\sum_{j=-k}^{j=k} \sum_{i=-k}^{i=k} I_{x+i,y+j} \tag{8}$$

$$sd(x, y) = \tfrac{r_{sd}}{n^2} (\textstyle\sum_{j=-k}^{j=k} \sum_{i=-k}^{i=k} I_{x+i,y+j}^2 - \tfrac{(sum(x,y))^2}{n}) \tag{9}$$

The normalised standard deviation based on a 3×3 window was successfully used in [31]. Therefore, the normalised variance nsd is also considered for f. The normalised variance nsd is defined in Equation (10). Here, the window size is $n \times n$. Note that the scale parameter r_{sd} in nsd can be different from sd.

$$nsd(x, y) = \frac{n^2 * sd(x, y)}{sum(x, y)} \tag{10}$$

When IFC is generated, one of the specific features f (d_x, d_y, g, sd and nsd) is randomly selected, and t is randomly generated in the range from 0 to 50. Note that we rescale the ranges of d_x, d_y, g, sd and nsd so that threshold t is expected to be effectively used for all of the specific features.

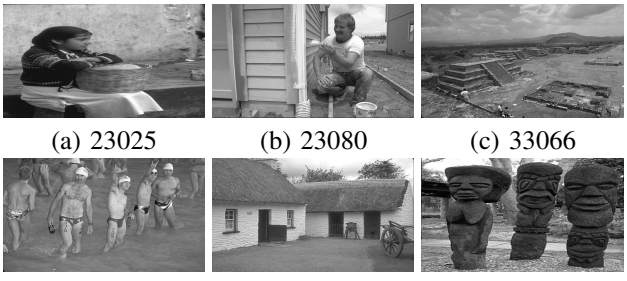
To construct different programs, functions $prog2(P1, P2)$ and $prog3(P1, P2, P3)$ are employed in the function set, where $P1$, $P2$ and $P3$ are sub-programs. In $prog2$ and $prog3$, all sub-programs are executed from left to right. For instance, $prog2(P1, P2)$ calls $P1$ firstly, then calls $P2$.

C. Fitness Function

Energy functions in active contours have been used to directly find a set of curves in an image without help from ground truth. However, the existing energy functions are usually based on the global edge information in a detected image. The computational cost for finding the minimum energies is very heavy. In order to quickly find edges using GP without ground truth, the fitness function in GP should be simplified from the energy functions used in active contours.

If an energy function is used for a local area only, such as a moving window, the computational cost of searching the minimum energy would be reduced obviously. A fitness function, similar to an energy function used in a moving window, was proposed to search edges in the moving window using Particle Swarm Optimisation [33]. The results [33] show that an energy function based on a local area can be helpful to find good edges. Therefore, the designed fitness function in GP will mainly address the edge information from local areas.

To relax the constrain in an energy function, the designed fitness function will allow broken edges and thick edges. The



(a) 23025 (b) 23080 (c) 33066

(d) 370036 (e) 385028 (f) 101085

Fig. 1. Five BSD training images (a)–(e) and one BSD test image (f).

new fitness function Fit is defined in Equation (12). Here, g_i is the image gradient for pixel i, N is the number of detected edge points, w_1 and w_2 are weight factors, and pw_i is a penalty weight for thickness. When pixel i (marked as an edge point) has more than four neighbours marked as edge points, pw_i is equal to 1, otherwise, pw_i is equal to 0. Note that EE returns positive infinity when N is equal to 0.

$$EE = \frac{1}{\log\left(\frac{\sum_{i=1}^{N} g_i}{N}+1\right)+1} + \frac{w_1}{\log\left(\sum_{i=1}^{N} g_i+1\right)} \quad (11)$$

$$Fit = EE + \frac{w_2 \sum_{i=1}^{N} pw_i}{N} \quad (12)$$

Different from $EE(c)$ in Equation (2), EE ranges from 0 to positive infinity, not from 0 to 1. Since the thickness is allowed in EE, the first part $\frac{\sum_{i=1}^{N} g_i}{N}$ in EE is used to look for edge points with high image gradients. The second part $\sum_{i=1}^{N} g_i$ is expected to find as many edge points as possible. When a high value of the first part in EE is obtained, pixels with low g would be not considered as edge points. Therefore, there is a trade-off between the first part and the second part.

IV. Experiment Design

This section describes the training images and test images used for the experiments. Also, the settings of the experiments are provided.

A. Image Dataset

The Berkeley Segmentation Dataset (BSD) [4] has been popularly used for edge detection and image segmentation. The BSD dataset provides ground truth for predefined training images and predefined test images, and the training and test images are independent. There are 200 training images and 100 test images. The ground truth are not used in the training stage in this paper. Note that each image has 481×321 pixels and comes from the natural world.

Fig. 1 shows five BSD training images and one BSD test image. Note that these images are rescaled in Fig. 1. These images are independently used to find GP edge detectors without ground truth. The reason for selecting these images is that these images have rich edge information and large numbers of true edge points.

B. Experiment Settings

The parameter values for GP are: population size 100; maximum generations 30; and probabilities for mutation 0.30, crossover 0.65 and elitism (reproduction) 0.05. The maximum depth (of a program) is 3. These values are set based on initial experiments. The reason for choosing a small depth is that evolved programs are mainly focused on the combination of different conditions in $IFC(f, t, P1, P2)$ and fast detection speed. It is possible to increase the computational cost for new programs when the depth is large. We perform 30 independent runs for each single image.

For the $n \times n$ moving window, the parameter n is selected 9 in the experiments. To rescale the range of sd and nsd, we use $r_{sd} = 8$ in sd and $r_{sd} = 10$ in nsd.

In fitness function Fit, w_1 is set to 80, and w_2 is set to 4. The parameters w_1 and w_2 are set based on initial experiments. How to choose w_1 and w_2 needs to be investigated in the future.

V. Results and Discussion

This section describes the results with discussions. Firstly, the results (Fit) on the six images are given. Secondly, the test performance of the GP edge detectors on the BSD test images is compared to the image gradient using a single threshold from single images. Thirdly, the computational cost of training and testing the GP edge detectors will be discussed. Fourthly, examples of detected images are provided. Lastly, examples of GP edge detectors are interpreted.

Note that the evolved programs from a single image will be compared based on the 100 test BSD images. The popular F-measure [4][34] is employed in the testing phase for performance evaluation. The F-measure $F = \frac{2\text{recall}*\text{precision}}{\text{recall}+\text{precision}}$ combines recall and precision. Here, recall is the ratio of the total number of correctly marked edge points to the total number of true edge points; and precision is the ratio of the total number of correctly marked edge points to the total number of marked edge points. More details about the F-measure technique can be found in [4].

A. Fitness Function Fit

Table I gives the final fitness values of the GP edge detectors and the minimum values from using a single threshold to obtain edges. We use $Thresh$ to indicate the results from a single threshold selected based on the minimum Fit. Here, t-tests are used for the comparisons. The p-values come from the comparisons between the GP edge detectors and the minimum values from a single threshold, and "$\downarrow$" indicates that the fitness values of the GP edge detectors are significantly lower (better) than the minimum fitness value from a single threshold. Giving that g_{max} is the maximum g in a given single image, we use a set of fixed thresholds $\frac{th*g_{max}}{51}$ $(th = 1, 2, \ldots, 51)$ to find the minimum Fit on the image gradient g of a given image. Here, 51 thresholds are considered to be sufficient to find the minimum fitness on the image gradient, and they are not used in GP.

As we can see, the results from the GP edge detectors on the six single images are significantly better than the results

TABLE I

FITNESS FUNCTION Fit VALUES (MEAN $\pm$ STANDARD DEVIATION) ON THE GP EDGE DETECTORS AND SINGLE THRESHOLDS

Training Image	GP	$Thresh$	p-values
23025	15.3742 ± 0.0366	28.6864	$0.0000 \downarrow$
23080	15.2990 ± 0.0036	28.1758	$0.0000 \downarrow$
33066	15.6053 ± 0.0104	28.7084	$0.0000 \downarrow$
370036	15.8087 ± 0.0219	29.2161	$0.0000 \downarrow$
385028	15.4535 ± 0.0393	28.7369	$0.0000 \downarrow$
101085	14.7406 ± 0.0416	27.2433	$0.0000 \downarrow$

from a single threshold, in terms of Fit. From the view of minimising Fit, GP can find better solutions than using a set of single thresholds on image gradient g. It is possible that the condition information in IFC are helpful for finding detectors with small Fit. Note that fitness function Fit cannot directly represent the detection performance F and whether the GP edge detectors found from these given single images can effectively detect other images. The test performance on the GP evolved edge detectors is discussed in the next subsection.

B. Test Performance F

Table II shows the test performance F and the comparisons between the results from GP edge detectors and the results from the Sobel edge detector, the image gradient g, and a single threshold based on the minimum Fit on a single image ($Thresh$). Here, the results from the Sobel edge detector and the image gradient g are selected based on the maximum F when thresholds $\frac{th}{51}$ ($th = 1, 2, \ldots, 51$) are used to obtained binary edges. The maximum performance F is 0.4832 for the Sobel edge detector, and $F = 0.5326$ for the image gradient g. Since these thresholds are selected based on the test BSD images, we also use thresholds selected from given single images to directly obtain results based on the image gradient g. A threshold from $\frac{th * g_{max}}{51}$ is directly used when the relevant binary edges in the given image has the minimum Fit. In the table, "$\downarrow$" indicates that the result from the relevant row is significantly better than the results from the GP edge detectors in a given image; and "$\uparrow$" indicates that the results from the relevant row is significantly worse than the results from the GP edge detectors. Here, t-tests are employed and the significance level is 0.05. Since only a single threshold is used in the image gradient g and the Sobel edge detector, there are only single F values from these detectors.

There are four interesting observations from Table II. Firstly, the table shows that the GP edge detectors have stable detection performances, although only single images are used to find the programs. Secondly, the best performance F from the image gradient g is significantly better than the results from the GP edge detectors, but the GP edge detectors from the five single training images have very close test performances (less than 0.0061) to the image gradient g in terms of F. Note that the image gradient g is normalised for the soft edge maps on the 100 test images. However, fixed thresholds are directly used to g without normalisation in

TABLE II

TEST PERFORMANCE F VALUES (MEAN $\pm$ STANDARD DEVIATION OF 30 RUNS) FOR THE GP EDGE DETECTORS AND SINGLE THRESHOLDS, AND p-VALUES FOR COMPARISONS.

Training Image	F		p-values		
	GP	$Thresh$	Sobel	g	$Thresh$
23025	0.5265 ± 0.0097	0.4904	$0.0000 \uparrow$	$0.0020 \downarrow$	$0.0000 \uparrow$
23080	0.5288 ± 0.0010	0.4865	$0.0000 \uparrow$	$0.0000 \downarrow$	$0.0000 \uparrow$
33066	0.5278 ± 0.0015	0.4814	$0.0000 \uparrow$	$0.0000 \downarrow$	$0.0000 \uparrow$
370036	0.5273 ± 0.0018	0.4762	$0.0000 \uparrow$	$0.0000 \downarrow$	$0.0000 \uparrow$
385028	0.5278 ± 0.0047	0.4560	$0.0000 \uparrow$	$0.0000 \downarrow$	$0.0000 \uparrow$
101085	0.5218 ± 0.0113	0.4986	$0.0000 \uparrow$	$0.0000 \downarrow$	$0.0000 \uparrow$

The results from $Thresh$ are based on the image gradient g using a single threshold with minimum Fit on a single image. The comparisons are between the results from the GP edge detectors and the results from the **Sobel** edge detector (**F=0.4832**), the **image gradient** g (**F=0.5326**) and $Thresh$.

the GP edge detectors. Since we use the image derivatives in the energy function, it is a potential reason that the evolved GP edge detectors (using fixed thresholds) cannot outperform than the best performance F from the image gradient g. Thirdly, a single threshold based on the minimum Fit on a given image is not good to use for unseen images. The results based on single thresholds on the given single images are significantly worse than the results from the GP edge detectors. For image 385028, the difference performance between the binary edges from a sing threshold and the binary edges from GP edge detectors is 0.0718. The GP edge detectors are significantly better than the Sobel edge detector, but using a single threshold based on the minimum Fit is not better than the Sobel edge detector. Lastly, the GP edge detectors from the test image 101085 can also detect the 100 test images with a stable good performance, in terms of F. It seems that we can use the GP system to directly evolve good GP edge detectors on a test image. And, the GP evolved edge detectors can be effectively used for other unseen images.

In summary, the GP system can use a single image to evolve good edge detectors for effectively detecting edges on the 100 test BSD images. However, a single threshold based on the minimum Fit from a single image is not good for the image gradient g to obtain binary edges on the test images. A potential reason for the GP edge detectors to have good performance is that the GP system not only uses image derivatives, but also variances. Via combining image derivatives and variances together, the GP system could evolve stable edge detectors so that these edge detectors can also effectively detect edges on other images. Note that the single images used in the experiments include rich edge information and large numbers of true edge points. Whether different types of images (such as including high contrast objects or only a few true edge points) affect the performance of evolved edge detectors will be investigated in the future.

C. Computational Cost

Table III gives the averages of training and test times on the GP edge detectors and the image gradient g on the 100 test BSD images. The p-values are from the comparisons

TABLE III

AVERAGE OF TRAINING AND TEST TIMES (IN SECONDS) OF THE GP EDGE DETECTORS AND THE IMAGE GRADIENT g ON THE 100 TEST BSD IMAGES.

	Training Time	Test	
		Time	p-value
23025	88.2253 ± 11.7024	0.0335 ± 0.0088	$0.0000 \downarrow$
23080	87.5247 ± 11.2661	0.0163 ± 0.0155	$0.0202 \uparrow$
33066	87.9527 ± 12.4387	0.0334 ± 0.0123	$0.0000 \downarrow$
370036	83.7397 ± 7.8237	0.0312 ± 0.0074	$0.0000 \downarrow$
385028	87.0747 ± 10.9157	0.0331 ± 0.0080	$0.0000 \downarrow$
101085	86.2603 ± 16.9021	0.0315 ± 0.0097	$0.0000 \downarrow$
g	$-$	0.0199 ± 0.0021	$-$

between the GP edge detectors from a single image and the image gradient g by using t-tests. From the table, GP only takes around 86 seconds to find a good edge detector based on a single image. The GP evolved edge detectors takes less than 0.04 seconds on a BSD test image (of size 321×481 pixels), which is suitable for real-time applications (less than 0.1 second).

Compared to the image gradient g, all GP evolved edge detectors from the six single images, except for image 23080, takes a significantly longer time to detect a test image. The GP evolved edge detectors from image 23080 are significantly shorter than the image gradient g. A potential reason is that some evolved edge detectors include the marker anE (using a 3×3 window) which has lower computational cost than markers m and nE (using 9×9 window). Also, the image gradient g includes normalisation, but the GP edge detectors do not do normalisation. From Table II and Table III, we can see that the GP edge detectors have good detection performance and fast detecting speed (shorter than 0.04 seconds).

D. Detected Images

Fig. 2 shows the binary results from the image gradient g on three example test images 101085, 106024 and 296007 from the BSD test image dataset. Here, "GT" is the ground truth for the three test images. The ground truth in the BSD image dataset are combined with the observations from five to ten people. The threshold (0.25) for these binary edge maps is selected based on the maximum F values. These detected binary edges are mainly focused on boundaries in high contrast areas (between objects and background). Edges in low contrast areas are filtered by the threshold (0.25).

Fig. 3 gives the binary results of the best GP edge detectors from single images on the three test images. Here, an image label in the first column is used to indicate the results from the GP edge detector trained by this image. From an overview, the differences among these binary results from the GP edge detectors and the results from the image gradient g are not very obvious. Compared to the image gradient g with the maximum F, the GP edge detectors detect more true edge points. For instance, in image 106024, the numbers of true edge points found by the GP edge detectors on the boundary of the animal are larger than the result from the image gradient g in Fig. 2. However, in images 101085 and

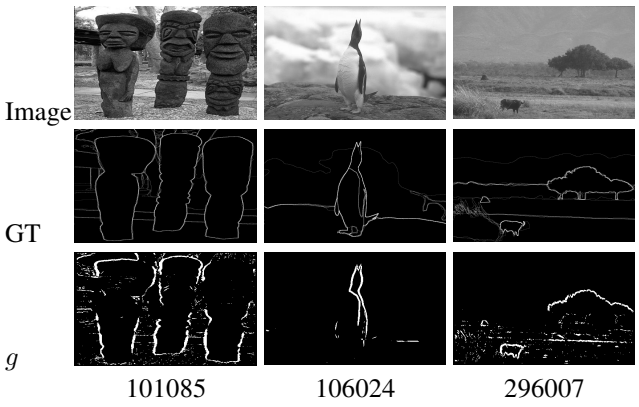


Fig. 2. Example detected images by the image gradient (with the maximum F on all test images).

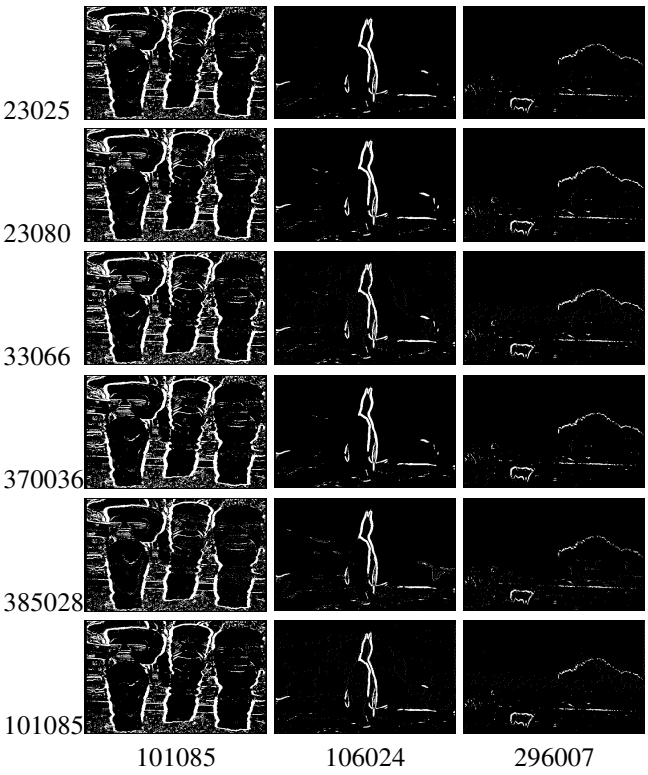


Fig. 3. Example detected images by GP edge detectors evolved from the single images (in the first column).

106024, the numbers of false edge points detected by the GP edge detectors are larger than the result from the image gradient g (with the maximum F).

From the detected edges on images 101085 and 106024, the thickness occurs in the detection results from the GP edge detectors. Different from active contours (thinned closed edges), fitness function Fit allows thick edges, and only uses a simple penalising weight pw_i (see Equation (12)). However, the detected results on image 296007 from the GP edge detectors are thinner than the detected edges on images 101085 and 106024. This different detection behaviour on the three test images might be caused by directly using fixed thresholds. Further investigation will be done in the future.

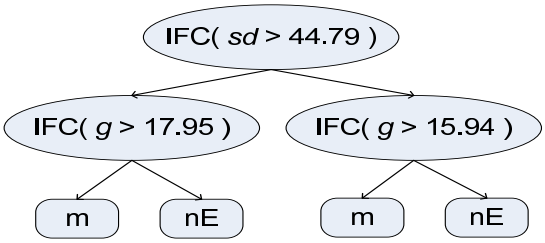


Fig. 4. Example GP edge detector gp_1 ($F = 0.5304$) with a common structure including IFC, sd, g, m and nE.

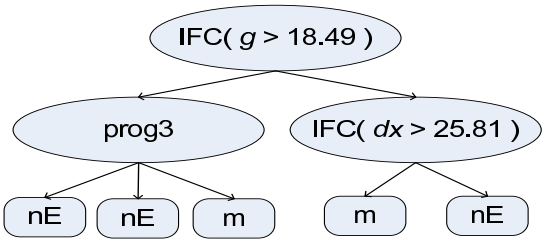


Fig. 5. Example GP edge detector gp_2 ($F = 0.5283$) with a structure including IFC, $prog3$, g, d_x, m and nE.

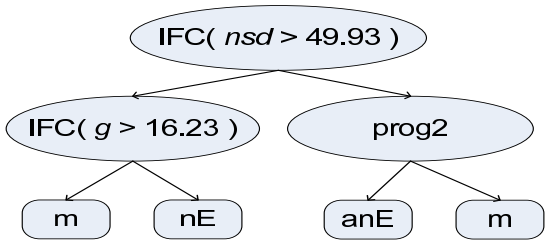


Fig. 6. Example GP edge detector gp_3 ($F = 0.5295$) with a structure including IFC, $prog2$, g, m, nE and anE.

E. Example GP Edge Detectors

The GP system mainly focuses on combinations of conditions in function IFC. The evolved GP edge detectors are analysed in this subsection. Three GP edge detectors are interpreted in the next paragraphs. The structures of all GP edge detectors are similar to these three GP edge detectors.

Fig. 4 shows a common structure in all GP evolved edge detectors. Note that node IFC will only call the left sub-program (child) if the condition in IFC is true, otherwise, node IFC only calls the right sub-program. For instance, in node $IFC(g > 17.95)$, the condition is $g > 17.95$. When the image gradient g in the current pixel is larger than 17.95, only the left sub-program is called.

From Fig. 4, both sub-programs of the root $IFC(sd > 44.79)$ are a common edge detector (using the image gradient to discriminate pixels as edge points or non-edge points). Note that the sub-programs use a fixed constant threshold on the image gradient g without normalisation. From the structure of GP edge detector gp_1, sd in the root is used to detect the variance of the neighbours around a discriminated pixel. If sd is high, the image gradient g might be affected by noise, and a high threshold for g is used to filter noise. If sd is low, the influence from noise should be low, and a low threshold for g is used. It seems that GP edge detector gp_1 uses an adaptive threshold on the image gradient g to discriminate pixels. The value of the threshold is chosen based on sd.

Fig. 5 gives another GP edge detector gp_2. GP edge detector gp_2 firstly marks pixels whose g values are larger than 18.49. For a pixel with not large g, it could be still considered as an edge point if its horizontal derivative d_x is high (larger than 25.81). GP edge detector gp_2 can be considered as an ordinary edge detector that utilises double thresholds (like a Canny edge detector [14]) to find pixels based on the image gradient g and the image derivative d_x. Note that node $prog3$ marks the pixel offset from the current pixel with a large g. Since the image gradient g has thickness responses on edges and the evaluation F allows that a detected edge point has limited offset distance from the true edge point [4], there is no obvious influence on test performance F from node $prog3$.

Fig. 6 shows GP edge detector gp_3. Here, the root $IFC(nsd > 49.93)$ performs a filtering function. In node $prog2$, if there is a small variance in the current discriminating area, a set of pixels in this area will be marked as non-edge points. After we move the current position to a new position, and the pixel in the new position will be marked as an edge point immediately. However, when the current position moves to the right neighbour, the neighbours of the new position will be marked as non-edge points if nsd in the new position is lower than 49.93. It is possible that the left neighbour of the new position, previously marked as an edge point is marked as a non-edge point. It seems that Node $prog2$ detects edges based on the normalised variances of a discriminated pixel and its neighbours.

From the three example GP edge detectors, those programs can be reasonably interpreted. Although the maximum depth of a GP program is small, different effective structures have been found by the GP system. Based on the three examples of GP edge detectors, the combination of different edge features, such as the image gradient and the variance, can be used to find good edge detectors based on single images. Since the features in the experiments mainly come from the image gradient, whether new features (such as the image histogram) can be included to improve detection performance needs to be investigated in the future.

In addition, there are two interesting observations from analysing the structures of all GP edge detectors. Firstly, the structure of $IFC(g, t, m, nE)$ (using the condition of whether the image gradient g is larger than threshold t) has very high occurrences in the evolved GP edge detectors. Almost of all GP edge detectors include $IFC(g, t, m, nE)$. Secondly, mH and mV are not found in all GP edge detectors. A potential reason is that mH and mV could falsely mark pixels (true non-edge points) as edge points. Also, it is possible that anE falsely marks pixels (true edge points) as non-edge points. From all GP edge detectors, there are only a few anE nodes. It seems that directly marking a set of pixels might be not good for extracting edges. Statistical techniques will help to analyse the structures of

GP evolved edge detectors in the future.

VI. Conclusions

The goal of this paper was to develop an unsupervised learning GP system with low computational cost for edge detection. A fitness function, modified from the energy functions in active contours, was used to select good GP edge detectors without ground truth. Six single images were separately used to evolve edge detectors by GP. The GP evolved edge detectors from the six single images were used to test a set of unseen images (the 100 test BSD images). From the results, the goal was successfully achieved.

The experiments show that GP can be used to evolve good edge detectors from a single image without ground truth. Those GP evolved edge detectors are better than the Sobel edge detector on the 100 test BSD images, and the test performance of those GP edge detectors is close to the best performance from the image gradient. Also, the results show that the GP edge detectors evolved from a single image can be used to effectively detect other unseen images, but a threshold selected based on the image is not good to detect other images. In addition, from the analysis of the structures of GP edge detectors, the GP evolved edge detectors can be reasonably interpreted.

From the detected images, the problem of detecting thick edges should be addressed in the future work. To improve detection performance, a more effective GP system will be also investigated in the future.

References

[1] G. Papari and N. Petkov, "Edge and line oriented contour detection: state of the art," *Image and Vision Computing*, vol. 29, pp. 79–103, 2011.

[2] M. Kunt, "Edge detection: a tutorial review," in *Proceedings of the IEEE International Conference on ICASSP*, 1982, pp. 1172–1175.

[3] M. Basu, "Gaussian-based edge-detection methods: a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 32, no. 3, pp. 252–260, 2002.

[4] D. Martin, C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, 2004.

[5] W. Fu, M. Johnston, and M. Zhang, "Genetic programming for edge detection: a global approach," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2011, pp. 254–261.

[6] ——, "Soft edge maps from edge detectors evolved by genetic programming," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2012, pp. 24–31.

[7] C. Harris and B. Buxton, "Evolving edge detectors with genetic programming," in *Proceedings of the First Annual Conference on Genetic Programming*, 1996, pp. 309–314.

[8] R. Poli, "Genetic programming for image analysis," in *Proceedings of the First Annual Conference on Genetic Programming*, 1996, pp. 363–368.

[9] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, pp. 321–331, 1988.

[10] T. Chan and L. Vese, "Active contours without edges," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266–277, 2001.

[11] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[12] E. Miller, N. Matsakis, and P. Viola, "Learning from one example through shared densities on transforms," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2000, pp. 464–471.

[13] L. Ganesan and P. Bhattacharyya, "Edge detection in untextured and textured images: a common computational framework," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 27, no. 5, pp. 823–834, 1997.

[14] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[15] J. M. S. Prewitt, "Object enhancement and extraction," *Proceedings of The IEEE*, 1970.

[16] W. Frei and C.-C. Chen, "Fast boundary detection: A generalization and a new algorithm," *IEEE Transactions on Computers*, vol. C-26, no. 10, pp. 988–998, 1977.

[17] C. Lopez-Molina, B. De Baets, and H. Bustince, "Quantitative error measures for edge detection," *Pattern Recognition*, vol. 46, no. 4, pp. 1125–1139, 2013.

[18] J. Illingworth and J. Kittler, "A survey of the Hough transform," *Computer Vision, Graphics and Image Processing*, vol. 44, pp. 87–116, 1988.

[19] J. Xu, O. Chutatape, and P. Chew, "Automated optic disk boundary detection by modified active contour model," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 3, pp. 473–482, 2007.

[20] C. Xu and J. Prince, "Snakes, shapes, and gradient vector flow," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 359–369, 1998.

[21] G. Aubert and P. Kornprobst, "Mathematical problems in image processing: Partial differential equations and the calculus of variations," *Applied Mathematical Sciences*, vol. 147, pp. 173–174, 2006.

[22] Y. Zhang and P. I. Rockett, "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*, 2005, pp. 795–802.

[23] T. Golonek, D. Grzechca, and J. Rutkowski, "Application of genetic programming to edge detector design," in *Proceedings of the International Symposium on Circuits and Systems*, 2006, pp. 4683–4686.

[24] G. Hollingworth, S. Smith, and A. Tyrrell, "Design of highly parallel edge detection nodes using evolutionary techniques," in *Proceedings of the Seventh Euromicro Workshop on Parallel and Distributed Processing*, 1999, pp. 35–42.

[25] E. Bolis, C. Zerbi, P. Collet, J. Louchet, and E. Lutton, "A GP artificial ant for image processing: preliminary experiments with EASEA," in *Proceedings of the 4th European Conference on Genetic Programming*, 2001, pp. 246–255.

[26] W. Fu, M. Johnston, and M. Zhang, "Genetic programming for edge detection: a global approach," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2011, pp. 254–261.

[27] I. Kadar, O. Ben-Shahar, and M. Sipper, "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*, 2009, pp. 1887–1888.

[28] W. Fu, M. Johnston, and M. Zhang, "Automatic construction of Gaussian-based edge detectors using genetic programming," in *Proceedings of the 16th European Conference on Applications of Evolutionary Computation*, 2013, pp. 365–375.

[29] M. I. Quintana, R. Poli, and E. Claridge, "Morphological algorithm design for binary images using genetic programming," *Genetic Programming and Evolvable Machines*, vol. 7, pp. 81–102, 2006.

[30] J. Wang and Y. Tan, "A novel genetic programming based morphological image analysis algorithm," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, 2010, pp. 979–980.

[31] W. Fu, M. Johnston, and M. Zhang, "Automatic construction of invariant features using genetic programming for edge detection," in *Proceedings of the 25th Australasian Joint Conference on Artificial Intelligence*, vol. 7691, 2012, pp. 144–155.

[32] ——, "Genetic programming for edge detection using multivariate density," in *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, 2013, pp. 917–924.

[33] M. Setayesh, M. Zhang, and M. Johnston, "Detection of continuous smooth and thin edges in noisy images using constrained particle swarm optimisation," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2011, pp. 45–52.

[34] P. Dollar, Z. Tu, and S. Belongie, "Supervised learning of edges and object boundaries," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 1964–1971.