Wenlong Fu
School of Engineering and Computer Science, Victoria University of Wellington, NEW ZEALAND

Bing Xue
School of Engineering and Computer Science, Victoria University of Wellington, NEW ZEALAND

Mengjie Zhang
School of Engineering and Computer Science, Victoria University of Wellington, NEW ZEALAND

Mark Johnston
Department of Mathematics, Institute of Science and the Environment, University of Worcester, UNITED KINGDOM

Fast Unsupervised Edge Detection Using Genetic Programming

Abstract

Edge detection has been a fundamental and important task in computer vision for many years, but it is still a challenging problem in real-time applications, especially for unsupervised edge detection, where ground truth is not available. Typical fast edge detection approaches, such as the single threshold method, are expensive to achieve in unsupervised edge detection. This study proposes a Genetic Programming (GP) based algorithm to quickly and automatically extract binary edges in an unsupervised manner. We investigate how GP can effectively evolve an edge detector from a single image without ground truth, and whether the evolved edge detector can be directly applied to other unseen/test images. The proposed method is examined and compared with a recent GP method and the Canny method on the Berkeley segmentation dataset. The results show that the proposed GP method has the ability to effectively evolve edge detectors by using only a single image as the whole training set, and significantly outperforms the two methods it is compared to. Furthermore, the binary edges detected by the evolved edge detectors have a good balance between recall and precision.

Digital Object Identifier 10.1109/MCI.2018.2866729
Date of publication: 15 October 2018

I. Introduction

Edge detection has been an active area of research over many years, and it is important for processing and understanding images [1]. Edges are the boundaries between different areas, such as background and objects, in digital images. Pixels on these boundaries are edge points. A binary edge point indicates its corresponding pixel in the image is an edge point or not. A binary edge map consists of a set of binary edge points. To find edge points, different approaches have been proposed [1–3]. These approaches can be categorized into supervised edge detection (with ground truth provided) or unsupervised edge detection (without ground truth). This work focuses mainly on unsupervised edge detection.

In unsupervised edge detection, searching for binary edge maps is difficult and computationally expensive [1]. Most existing methods have high recall (i.e. the number of true edge points being detected over the total number of true edge points), but

Corresponding Author: Wenlong Fu
(Email: wenlong.fu@gmail.com)

1556-603X/18©2018IEEE

low precision (i.e. the number of detected true edge points over the total number of detected points), being easily affected by noise [4, 5]. The zero threshold technique proposed in [6] normally obtains binary edge maps with high precision but low recall. When both the recall and precision measures are considered, we need a further investigation on how to balance them. Generally, an optimization method is used to determine a threshold for each image. However, a threshold found for an image might not be good for other images. Instead of directly searching for a threshold for each image, it is desirable to investigate whether a detector trained from an image can be directly used for other unseen images. In one-shot learning [7], prior knowledge from existing datasets are utilized to quickly obtain classifiers for recognizing objects. A very small set of training images is used in one-shot learning algorithms. In our previous work [8], a single training image can be effectively used to evolve edge detectors by Genetic Programming (GP) [9]. GP is a population based evolutionary computation method, where each individual is a candidate solution of the target problem and is often represented as a *tree* or computer *program* [9]. GP has been employed to automatically evolve edge detectors since 1996 [10, 11]. However, there have been only very few works using GP in *unsupervised* edge detection to date [12]. Therefore, it is desirable to investigate how to employ GP to evolve edge detectors from a single image in an unsupervised manner and how the evolved edge detectors can be used to detect binary edge maps on unseen images.

Unsupervised edge detection is challenging and computationally expensive. Instead of searching for edge points in each image, it is desirable to quickly learn an edge detector that can be used for many images.

The overall goal of this paper is to investigate unsupervised edge detection using GP to improve detection precision without adversely affecting recall while keeping a low computational cost. A GP system is proposed to evolve edge detectors, represented as GP programs or trees, for marking edge points in an image. Instead of using a simple human-designed rule for marking edge points (e.g. if results are larger than a threshold, the pixels will be marked as edge points), such rules will be automatically evolved by the proposed GP system. To improve the efficiency, only a single image without ground truth is used as the whole training data for evolving edge detectors, and the evolved edge detectors are directly applied to extract binary edge maps for unseen images. Specifically, the following research objectives will be investigated:

- ❑ Whether a single training image without ground truth can be used to evolve good edge detectors by GP for directly detecting edges in unseen images,
- ❑ Whether the automatically generated rules by the proposed method are better than the GP system using fixed rules with a single threshold in [12],
- ❑ Whether the evolved edge detectors can obtain better detection performance than the commonly used Canny edge detector [13], and
- ❑ Whether the evolved rules for marking pixels are flexible and adaptable to different images.

In the remainder of this paper, Section II gives background on edge detection and GP. Section III describes the proposed GP system. After giving the settings of the experiments in Section IV, Section V provides the results with discussions. Section VI draws conclusions and suggests future research directions.

II. Background

This section briefly describes the background on edge detection, mainly unsupervised edge detection, thresholding techniques, and existing work using GP for edge detection.

A. Edge Detection

Edge detection often includes three stages: pre-processing, feature extraction, and post-processing [1, 2]. In unsupervised edge detection, there is no ground truth to train edge detectors. Noises are expected to be filtered in the pre-processing stage, and then edge features are extracted. For example, differentiation exists among different boundaries, and differentiation-based edge detection is used to obtain edge responses (edge features) on these boundaries [1–3, 14, 15]. After edge features are extracted, thresholding techniques are usually employed in the post-processing stage [1, 13, 16, 17].

1) Supervised Edge Detection

In supervised edge detection, a trained edge detector normally has a fast detection speed on unseen images [18]. A single fixed threshold is often used for the unseen images to determine pixels as edge points or non-edge points [6]. The computational cost of obtaining binary edge maps needs to be minimized for real-time applications. However, it is often hard for most existing methods to obtain a full binary edge map from an image (normally larger than 256×256 pixels) within a short time. In video processing, the time of processing each frame is often required to be shorter than 0.1 second [19]. Otherwise, it can easily cause the flicker problem. To quickly obtain binary edge maps, the computational cost needs to be low on both edge feature extraction and final binary edge points determination. In our previous work [20], training images and their ground truth were used to automatically evolve good low-level edge detectors with a low computational cost. When there is no ground truth, an unsupervised edge detector generally extracts a binary edge map based on a set of thresholds, instead of a single fixed threshold, but it is difficult to determine values of these thresholds, since the evaluation function for balancing edge points and non-edge points is difficult to design [5, 6, 17].

2) Unsupervised Edge Detection

There are generally three approaches to unsupervised edge detection: searching for thresholds [6, 17], finding connected curves [15, 21, 22], and obtaining boundaries based on segmentation results [23]. Thresholding techniques normally obtain edge points pixel by pixel. Hence, they are generally fast and suitable for real-time applications. However, such techniques consider very little global information and usually do not balance between the edge points and non-edge points. For example, an iterative method and an entropy based thresholding technique were employed to obtain binary edge maps in [17], and the Otsu method [24] was utilized to search for good thresholds to obtain edge maps [17]. However, the detected results from these three methods could be easily affected by noise. In addition, the binary edge maps from the thresholding techniques in [4] and [5] typically have low precision. The zero threshold technique proposed in [6] normally obtains binary edge maps with low recall. However, both recall and precision are important, since recall cannot be the proportion of non-edge points that are incorrectly detected as edge points, but precision cannot indicate the proportion of true edge points being missed. When both recall and precision are considered, how to balance between them needs further investigation. Approaches based on finding connected curves or obtaining boundaries from segmentation results usually consider context [1] in a detected image. The active contour approach [21, 25] utilizes an energy function to find good closed edges. However, these methods [1, 21, 25] suffer from high computational cost. Generally, only parts of the edges in an image can be found by this approach. Also, binary edges are usually dependent on initial candidate curves [1, 26]. Since an energy function considers edge curves, instead of independent individual edge points, it is worth investigating how to integrate an energy function with a thresholding technique to take the advantages of both approaches.

Image gradients are popularly used to extract edge features in unsupervised edge detection, such as the Sobel edge detector [27] and the Canny edge detector [13]. A common computational framework is suggested in [27] to calculate gradients on untextured and textured images. In general, a horizontal derivative and a vertical derivative are combined as the image gradient, which are also used to obtain edge orientation information. To filter noise, Gaussian filters have been applied to edge detection [2, 13]. Considering the effect of white noise, the Canny detector employed a Gaussian filter to approximate a given function (considered as an optimal edge detector) [13]. In a two-dimensional Gaussian filter $g_\sigma(x, y)$ (see Eq. (1)), the image horizontal derivative $\partial g(x, y)/\partial x$ and the image vertical derivative $\partial g(x, y)/\partial y$ are defined in Eqs. (2) and (3), respectively. Here, σ is a scale parameter. Gaussian image gradient $\nabla g(x, y)$ is defined in Eq. (4), and the edge direction θ is defined in Eq. (5).

$$g_\sigma(x, y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (1)$$

$$\frac{\partial g(x, y)}{\partial x} = -\frac{x}{2\pi\sigma^4}\exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (2)$$

$$\frac{\partial g(x, y)}{\partial y} = -\frac{y}{2\pi\sigma^4}\exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (3)$$

$$\nabla g(x, y) = \sqrt{\left(\frac{\partial g(x, y)}{\partial x}\right)^2 + \left(\frac{\partial g(x, y)}{\partial y}\right)^2} \quad (4)$$

$$\theta = \arctan\left(\frac{\frac{\partial g(x, y)}{\partial y}}{\frac{\partial g(x, y)}{\partial x}}\right). \quad (5)$$

There are many other methods for edge feature extraction [1], such as the image histogram gradient [28], the image texture gradient [28], techniques using the co-occurrence matrix [29], and automatic construction of edge features [30, 31]. Gaussian image gradient is popularly used for edge detection, and has stable performance on edge detection [1, 2]. It is also computational less expensive than other techniques, such as the image histogram gradient. Therefore, this paper utilizes only Gaussian image gradient to extract edge features.

3) Performance Evaluation

Since edge detection is subjective, i.e., different people may mark/label different observations of edges in the same natural images, it is not straightforward to evaluate binary edge maps, where natural images mean images coming from different natural scenes, such as a picture of a tree, an animal or a person on the grass. It is expensive to manually check the detected results. The ground truth of a set of images is often utilized to automatically evaluate the detected results [28, 32]. F-measure, which combines recall r and precision p with a parameter α, has been popularly used to measure the detection performance [28, 33]. F-measure is defined in Eqs. (6)-(8), where T is the number of true edge points, TP is the number of true edge points being correctly detected, P is the total number of predicted edge points, and parameter α is from 0 to 1. When α is large, F is mainly affected by r. For a soft edge map, a low threshold is usually used to obtain high r, but low p; and a high threshold is usually used to obtain low r, but high p. α is set to 0.5 to balance recall r and precision p [28, 33]:

$$r = \frac{TP}{T} \quad (6)$$

$$p = \frac{TP}{P} \quad (7)$$

$$F = \frac{rp}{\alpha r + (1 - \alpha)p}. \quad (8)$$

B. Thresholding Techniques

Thresholding techniques in image processing generally include histogram-based methods [16, 34] and spatial methods [17]. Histogram-based methods employ the histogram of edge responses, such as the image gradient, to search for a threshold value to divide the edge responses into different parts. In [16], an optimal threshold was selected from a set of pre-defined thresholds based on the Otsu method or the maximum entropy. Spatial methods directly apply a given threshold to edge responses for marking pixels as edge or normal points [18].

1) Thresholding in Supervised Edge Detection

When the training images and their ground truth are provided, a fixed threshold can be directly used. In our previous work, a fixed threshold of 0 was used in the edge detectors evolved by GP [18].

Different from using a given threshold, the probabilities of a pixel being an edge point or a non-edge point have been used to obtain binary edge maps [20, 35].

Existing unsupervised edge detection methods often have high recall, but low precision and are easily affected by noise.

2) Thresholding in Unsupervised Edge Detection

When there is no ground truth provided, results from different edge detectors can be combined to form ground truth [36], but it is computationally expensive. However, the detection time for a single image is required to be shorter than 0.1 second for many real-world problems. Therefore, thresholding techniques focus mainly on searching threshold values after obtaining a histogram on the edge responses, which are aggregated in a histogram with bins $k = 0, 1, \ldots, L-1$, where k is a threshold level and L is the number of threshold levels. The Otsu method [24] has been widely applied to histogram-based thresholding techniques [16, 34]. The Otsu method aims to separate pixels into edge points and non-edge points according to the minimum intra-class variance or (equivalently) the maximum between-class variance. An entropy-based thresholding technique [17] was used to find thresholds with the maximum entropy, where it is found that the Otsu method and the entropy-based thresholding technique discriminate pixels with high recall, but they are easily affected by noise.

Generally, the number of edge points and the number of non-edge points are unbalanced, and the number of true edge points is much smaller than that of non-edge points. Edge detection is often treated as a binary classification problem, and the edge point is the more important class. In [6], there was an assumption that the number of edge points is not less than 1% of the total pixels in an image based on their experimental experience. Since the numbers of true edge points in different images are very different, this method may lead to the further loss of some true edge points for images that contain very few true edge points.

3) Thresholding in Canny

In the Canny edge detector, a thresholding technique has been used [13] to eliminate breaking edge contours. A user-defined high threshold is used to mark pixels as strong edge points. After getting strong edge points, a user-defined relatively low threshold is used to mark a pixel as a weak edge point, which is considered as an edge point if it is connected to a strong edge point. However, it is expensive to manually tune the low and high thresholds in the Canny edge detector. In [34], a high threshold was determined based on the histogram on edge responses, and a low threshold was determined based on the distributions of edge points and non-edge points. In [4] and [5], a unimodal thresholding technique on the edge response histogram was proposed to find a low threshold. In the unimodal thresholding technique, the information (distribution of edge response magnitudes) from each threshold level is calculated. Therefore, the computational cost is high. The results in [5] show that these threshold techniques have high recall, but are affected by noise.

Overall, existing thresholding techniques focus mainly on recall (generally using low thresholds) or precision (generally using high thresholds) only. When both recall and precision are considered, thresholding techniques need to be further investigated to obtain binary edge maps in unsupervised edge detection.

C. Related Work on GP for Edge Detection

GP has been applied to supervised edge detection when the training data with desired outputs are provided. In low-level supervised edge detection, GP has been used to automatically design edge detectors based on pixel intensities. There are three ways to provide training data for evolving edge detectors. Firstly, the ground truth of training images is given by humans. The ground truth used in [37] is hand-labelled, and the ground truth used in [28] is labelled based on the segmentation results. Note that the segmentation results are determined by humans. Via selecting pixels from a 13×13 moving window to construct GP programs/trees, a multi-objective GP system is used to extract edge features [37]. GP has also been used to evolve edge detectors where pixels in a moving window were considered as terminals, i.e. the leaf nodes of GP trees/programs [38, 39]. To avoid setting a window size, search operators, such as a shifting operator, have been used as functions, i.e. the internal nodes of GP programs, to evolve edge detectors based on full images [11, 20]. Secondly, the "ground truth" of training images comes from existing edge detectors, such as the approximation of the Sobel detector [40] and the Canny detector [41]. Thirdly, when edges are considered as signals, GP is used to evolve formulae to approximate the designed "signal" responses on edges and non-edges points. One-dimensional step edge responses are designed for evolving formulae, which are used as edge detectors [10].

The GP evolved low-level edge detectors can compete well with the results from the existing edge detectors, such as the Sobel edge detector [18] and the Canny edge detector [37] according to F-measure. Also, specific domain knowledge for edge detection was used to evolve edge detectors by GP. For example, morphological erosion and dilation were used as terminals of GP trees to evolve edge detectors in binary images [42, 43]. Gaussian filters were used to evolve Gaussian-based edge detectors by GP [44, 45]. Statistical knowledge has been used to construct composite features in our previous work [35]. After utilizing specific domain knowledge in GP, the detection performance of the evolved edge detectors has been improved.

In summary, most of the existing works using GP for edge detection are based on ground truth. Our previous

With the flexible tree based representation, Genetic Programming is able to use different functions and terminals to evolve rules as edge detectors.

work conducted an initial investigation on unsupervised edge detection [12]. The results show that GP has the potential to evolve edge detectors from a single image. In this work we will further investigate the capability of GP for evolving edge detectors without ground truth.

III. The New Method: Modified GP (MGP)

This section introduces the proposed unsupervised edge detection method, a modified GP system (MGP), which is extended from our preliminary investigation on GP for unsupervised edge detection in [12]. This section firstly briefly describes the baseline algorithm in [12] which is called a GP artificial ant system (GPA), then introduces the new MGP method.

A. The Baseline Algorithm: GPA

GP [12, 39] has been used to design artificial ant sittings in an image to search for edge points without using ground truth. Edge points are considered as ant food sources. An action "eat" is used to mark a pixel as an edge point or not. GPA was proposed to evolve ants to search for edge points.

The *terminal* set in GPA includes four different types of markers: marking a pixel as an edge point, marking a set of pixels as edge points, marking a pixel as a non-edge point, and marking a set of pixels as non-edge points. The *function* set in GPA includes $\{IFC, NIF, +, -, \times, /, prog2(P1, P2)$ and $prog3(P1, P2, P3)\}$. In $IFC(f, t, P1, P2)$, f is a specific feature, t is a constant threshold, and $P1$ and $P2$ are sub-programs. $P1$ is executed if $f < t$; otherwise, $P2$ is executed. Considering the global information from the whole image and the local curves in a small moving window, a *fitness function* Fit_{GPA} shown in Eq. (9) was proposed for GPA, which relaxed the edge continuity constraint in an energy function:

$$Fit_{GPA} = EE + \frac{w_2}{N}\sum_{i=1}^{N} pw_i \qquad (9)$$

where

$$EE = \frac{1}{\log\left(\frac{1}{N}\sum_{i=1}^{N} g_i + 1\right) + 1} + \frac{w_1}{\log\left(\sum_{i=1}^{N} g_i + 1\right)} \qquad (10)$$

where N is the number of pixels marked as edge points, g_i is the image gradient for pixel i, w_1 and w_2 are weight factors, and pw_i is a penalty weight for thickness. A thick edge point for pixel i used $pw_i = 1$, which is only considered when the corresponding pixel is marked as an edge point and the number of its neighbors being marked as edge points is not smaller than five. For all other cases, pw_i is equal to 0. Function EE (Eq. (10)) includes the average of the image gradients $(1/N\sum_{i=1}^{N} g_i)$ and the sum of the image gradients $(\sum_{i=1}^{N} g_i)$.

Fit_{GPA} could utilize energies to effectively evaluate programs on a single image, but it has three parameters w_1, w_2, and pw_i to adjust. Further analysis shows that most of the evolved programs have similar structures and include mainly *IFC*. The terminal for marking a set of pixels as edge points has a very low usage over all the evolved programs. To effectively evolve edge detectors and reduce the number of the parameters in Fit_{GPA}, MGP is proposed in this paper.

B. Terminal Set

Table 1 lists the *terminal* set used in MGP and GPA, where the terminal set of MGP consists of $\{m, nE, anE, rnd, sd, g\}$. Each GP program scans pixels from left to right and from top to bottom in an image. To mark a pixel, we designed terminals called markers in GPA, as shown in Table 1. Two basic markers m and nE are utilized to mark pixels as edge points or non-edge points, respectively. Marker anE marks a set of pixels as non-edge points, i.e. the pixels within a small area.

Since the image gradient g was used in GPA without post-processing, there were thick responses on detected edges. When the moving window size is increasing, GPA might not handle thick responses well, and a large moving window size does not suit GPA. To thin edge responses, *non-maximum suppression* [13] is integrated into MGP. The terminal set of MGP includes the following two parts.

First, MGP keeps all the terminals from GPA except for the markers mH and mV, which were used to mark a set of pixels either horizontally or vertically. From the initial investigation, it was found that markers mH and mV were seldom selected in the evolved edge detectors. For a horizontal edge line, the width of a detected line might be two pixels if the non-maximum suppression is not used. If the non-maximum suppression is used, the width would be only one pixel. Therefore, using mH and mV may increase the false alarms.

TABLE 1 Terminals in MGP and GPA [12].

TERMINAL	MGP	GPA [12]	NOTE
m	YES	YES	MARK A PIXEL AS AN EDGE POINT
nE	YES	YES	MARK A PIXEL AS A NON-EDGE POINT
anE	YES	YES	MARK A SET OF PIXELS AS NON-EDGE POINTS
mH	NO	YES	HORIZONTALLY MARK PIXELS AS EDGE POINTS
mV	NO	YES	VERTICALLY MARK PIXELS AS EDGE POINTS
rnd	YES	NO	RANDOM CONSTANT
sd	YES	NO	STANDARD DEVIATION
g	YES	NO	GRADIENT

Second, random constants *rnd*, the image gradient *g*, and the standard deviation *sd* are used as terminals to construct conditions. $\{rnd, sd, g\}$ in GPA were used as arguments of function *IFC*, not individual terminals. *IFC* is still used in MGP. Fig. 1 provides a simple GP tree using function *IFC*. The tree is $IFC(0.2, g, m, nE)$, where g is the value generated by the non-maximum suppression operation on the image gradient from the current moving window and the condition is $0.2 < g$. These terminals $(rnd, g \text{ and } sd)$ return real numbers. They are used to combine sub-programs as conditions for calling different markers. Note that there are two different types of terminals, where each marker conducts an action, but the others $(rnd, g \text{ and } sd)$ return real numbers.

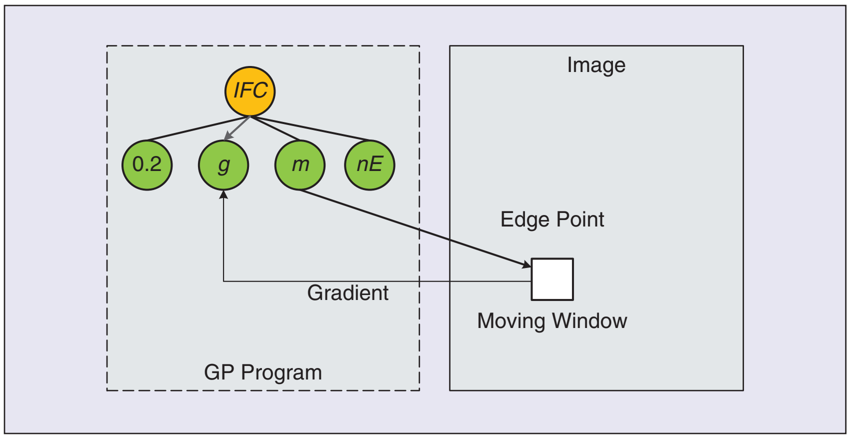

FIGURE 1 An example GP program for detecting edge points using a moving window.

C. Function Set

The conditions (rules) used in the functions (IFC) in GPA were based on a single fixed threshold. *IFC* was inspired by the experiential design of humans developing edge detectors to find edge curves, where the rules used are very limited. It is possible to evolve rules to improve detection performance. Therefore, MGP is designed to automatically generate sub-programs as conditions (rules). Since the directional markers mH and mV are not used in MGP, programs evolved by MGP search mainly for single edge points. Without directionally marking pixels, functions $prog2(P1, P2)$ and $prog3(P1, P2, P3)$ used in GPA are not included in MGP. Different from GPA using the given conditions, MGP automatically evolves conditions and edge detectors at the same time. Therefore, the conditions used in *IFC* are relaxed, and f and t are replaced by sub-programs *NP*1 and *NP*2 with the numerical return type. Here, *NP*1 and *NP*2 are constructed by the numerical terminals and arithmetic functions $\{+, -, \times, /\}$ used in MGP. Note that / is the protected division, returning 1 when being divided by 0.

In order to use the logical operator *IFC* in numerical return sub-programs (for automatically constructing conditions), a new numerical return function $NIF(NP1, NP2, NP3, NP4)$ is introduced in MGP, where *NP*3 and *NP*4 are sub-programs whose return types are numerical. When $NP1 < NP2$, *IFN* returns a numerical result from *NP*3; otherwise, a numerical result is obtained from *NP*4.

In summary, the function set in MGP consists of $\{IFC, NIF, +, -, \times, /\}$. Given the different return types of the terminals and functions, strongly typed GP [9] is used to develop the proposed MGP system.

D. Fitness Function

Since the non-maximum suppression is used, the thickness will not be considered. Fitness function Fit_{MGP} in Eq. (11) is actually the energy function *EE* shown by Eq. (10):

$$\begin{aligned} Fit_{MGP} &= EE \\ &= \frac{1}{\log\left(\frac{1}{N}\sum_{i=1}^{N} g_i + 1\right) + 1} \\ &\quad + \frac{w_1}{\log\left(\sum_{i=1}^{N} g_i + 1\right)}. \end{aligned} \tag{11}$$

There is a trade-off between the average of the image gradients $(1/N\Sigma_{i=1}^{N} g_i)$ and the sum of the image gradients $(\Sigma_{i=1}^{N} g_i)$. The average is high (low) when pixels with high (low) image gradients are selected. The sum is high (low) when a large number (or only a few) pixels are selected. In general, if the average is high, precision will be high because pixels with high gradients usually are true edge points. If the sum is high, recall is usually high because most pixels are marked as edge points. MGP removes the penalty item $w_2/N\Sigma_{i=1}^{N} pw_i$ in Fit_{GPA} in Eq. (9). There is only one parameter $w1$ in fitness function Fit_{MGP} in Eq. (11). $w1$ is used to balance recall and precision of the detected results, which has a similar function to α in F-measure in Eq. (8).

E. Unsupervised GP vs Thresholding Techniques

MGP is expected to evolve programs from a single image without ground truth, and the evolved program, i.e. edge detector, can be directly applied to extract edges from unseen images, i.e. images in the test set. Once the edge detector is evolved from the single image, the GP system does not need to restart to evolve a new edge detector again for unseen images. In contrast, a thresholding technique, such as the histogram-based method [16, 34], searches for an optimal threshold for each image and needs to restart the search for a threshold when the detected image is changed. If there are N images, there will be N optimization tasks of searching for optimal thresholds. Therefore, the proposed MGP system has low computational cost to detect edges on unseen images. The cost of the edge detectors evolved by MGP is mainly from the image gradient calculation.

IV. Experiment Design

A. Image Dataset

The Berkeley Segmentation Dataset (BSD) [28] is used in the experiments. There are 200 training images and 100 test images in BSD, and each image has 481×321 pixels or 321×481 pixels. The BSD dataset provides ground truth, but it is not used by MGP during the evolutionary learning process.

Six images shown in Fig. 2 are selected as training images, where five images are from the BSD training set and the other one (image 101085) is from the BSD test set (the ground truth of image 101085 is not used, i.e. unseen, during the training process). These images are chosen because they have rich edge information (such as intensity differences between objects and background) and relatively large numbers of true edge points.

We have conducted six sets of experiments, each using a different single image from Fig. 2 to evolve an edge detector. Each of the evolved edge detectors is evaluated on the 100 test images from the BSD dataset.

B. Experiment Settings

Table 2 lists the settings of MGP and GPA. MGP automatically evolves conditions to search over edge pixels, and the depth of an evolved sub-program as a condition could be larger than 1. Therefore, the maximum depth of a program in MGP should be larger than in GPA. However, since only *rnd*, *sd*, and *g*, and a small set of functions are used for generating rules (conditions), it is expected that sub-programs (used for these rules) are not too much larger. Therefore, the maximum depth is set to five in MGP. Based on initial experiments, a population size of 50 and the maximum number of generations is set to 30. Probabilities used for mutation (0.35), crossover (0.60) and reproduction (0.05) are taken from [46]. The initial population is created by the ramp-half-and-half method. 30 independent runs are conducted for each experiment.

For the $n \times n$ moving window, the parameter n is set to 11 in the experiments, since the best test F-measure performance of the Canny edge detector, i.e $F = 0.56$, is achieved when $n = 11$. The best test F-measure of $F = 0.56$ is the same as reported in [28]. Since GPA has a problem with thick/many responses and a large window size might not be suitable in GPA, we still keep $n = 9$ for GPA. Generally, at least one percent of the pixels in an image would be suggested as edge points [6]. Therefore, to approximately balance the two terms in Fit_{MGP}, we use $w_1 = 1/0.01 = 100$ in this paper.

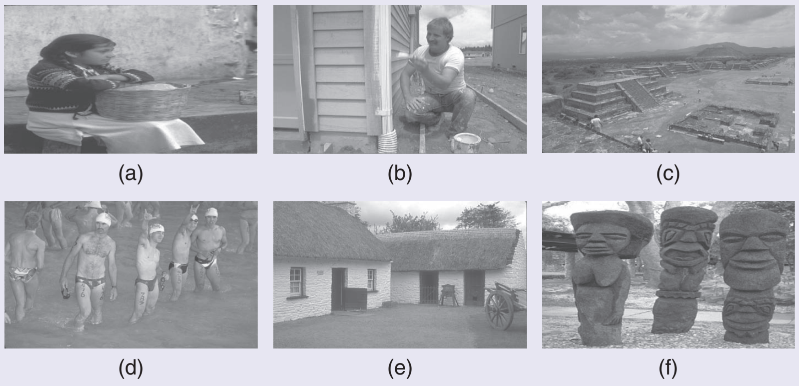

FIGURE 2 Five BSD training images (a)–(e) and one BSD test image (f). (a) 23025, (b) 23080, (c) 33066, (d) 370036, (e) 385028, and (f) 101085.

TABLE 2 Experiment Settings of MGP and GPA [12].

	MGP	GPA [12]
TERMINALS	$\{m, nE, anE, rnd, sd, g\}$	$\{m, mH, mV, nE, anE\}$
FUNCTIONS	$\{IFC, NIF, +, -, \times, /\}$	$\{IFC, prog2, prog3\}$
FITNESS FUNCTION	Fit_{MGP}	Fit_{GPA}
POPULATION	50	100
GENERATION	30	30
MUTATION	0.35	0.35
CROSSOVER	0.60	0.60
DEPTH	5	3
n (WINDOW SIZE)	11	9

V. Results and Discussion

The F-measure [28] is employed to evaluate the performance of MGP on the 100 test images in the BSD dataset.

A. Test Performance in terms of F-measure, Recall and Precision

Table 3 presents the means and standard deviations of the performance F values over the 100 test images for the edge detectors evolved by MGP and GPA over the 30 runs. The thick binary edges obtained from GPA were thinned by a binary thinning operator [47], then the thinned edges were used to obtain recall and precision. Two sample t-tests and Mann-Whitney-Wilcoxon (MWW, non-parametric) tests [48] with a significance level of 0.05 are used to compare the performance of MGP and GPA, where the p-values are presented in the third and fourth columns of Table 3, respectively. "↑" indicates that MGP is significantly better than GPA.

Table 3 shows that the test performance F of the edge detectors evolved by MGP is significantly better than that of GPA in all cases, i.e. using each of the six images as the training set. The overall test performance of MGP is, of course, a significant improvement over GPA, i.e.

from the average of 0.5267 to 0.5673. This suggests that by employing the new terminal set, function set and fitness function, MGP can further improve the performance of GPA in terms of the F-measure. We further investigate their performance in terms of the recall and precision in Table 4.

Table 4 gives the averages of recall and precision of the 30 edge detectors evolved by MGP or GPA (from the 30 runs) on the test set, when each of the six single images is used as the training data for evolving GP detectors. The MWW tests with a significance level of 0.05 are used to compare their performance. As can be seen from Table 4, the precision of the evolved edge detectors is significantly improved when MGP is used in all the six cases, from (a) to (f). Their improvement is over 0.58 or 13%, except for the 9.6% improvement when image (f) is used. The overall average is increased from 0.4607 to 0.5267 by around 0.066 or 14.33%. The results of recall in MGP are generally similar to GPA, with two significant smaller cases and four similar or better cases. The overall average recall is slightly decreased by MGP compared with GPA, from 0.6199 to 0.6180, by a very small value. Additionally, on training image (a), there is no significant differences between MGP and GPA in terms of recall.

According to Table 3 and Table 4, MGP mainly improves the precision of the detected results. There are two potential reasons: First, by using non-maximum suppression in MGP, the fitness function Fit_{MGP} does not need to evaluate the thickness of binary edges, and only addresses recall and precision using a weight w_1. However, fitness function Fit_{GPA} needs to address recall, precision, and the thickness of edges using three parameters w_1, w_2, and pw_i (see Eq. 9). It is more complicated to use Fit_{GPA} than Fit_{MGP} for evaluating edge detectors. Second, MGP automatically evolves conditions while evolving edge detectors, whereas in GPA, a limited set of conditions are pre-defined. The number of potential conditions evolved by MGP is larger than by GPA, and the evolved conditions are potentially better than the limited set of predefined conditions.

B. GP vs Canny

From [28], the best F performance of the Canny edge detector on the BSD test set is 0.56. The results of MGP from Table 3 are significantly larger than 0.56, according to the t-tests with the significance level of 0.05 and MWW tests. Note that the standard deviations of F values from MGP in Table 3 are very small. Most of the MGP edge detectors have detection performance F higher than 0.56. The image gradient used by the Canny edge detector is normalized (from 0 to 1). As discussed in Section II, since it is not easy to search for two optimal thresholds in the Canny edge detector, a high threshold is used to find strong edge points (with high magnitudes of the image gradient) and a low threshold is used to find weak edge points which are connected to strong edge points. If the high threshold is too large, some important edge points will be removed and the connected weak edge points will not be found. If the

TABLE 3 Mean ± standard deviation of the 30 *F* values on the test set by MGP and GPA. Note that *p*-values are from *t*-tests and Mann-Whitney-Wilcoxon (MWW) tests.

TRAINING IMAGE	MGP	GPA	*p*-VALUE (*t*-TEST)	*p*-VALUE (MWW)
(a)	0.5671 ± 0.0018	0.5265 ± 0.0097	0.0000 ↑	0.0000 ↑
(b)	0.5669 ± 0.0036	0.5288 ± 0.0010	0.0000 ↑	0.0000 ↑
(c)	0.5697 ± 0.0022	0.5278 ± 0.0015	0.0000 ↑	0.0000 ↑
(d)	0.5706 ± 0.0013	0.5273 ± 0.0018	0.0000 ↑	0.0000 ↑
(e)	0.5647 ± 0.0032	0.5278 ± 0.0047	0.0000 ↑	0.0000 ↑
(f)	0.5648 ± 0.0054	0.5218 ± 0.0113	0.0000 ↑	0.0000 ↑
average	0.5673	0.5267	–	–

TABLE 4 Recall and precision for the GP edge detectors evolved by MGP and GPA.

TRAINING IMAGE	MGP		GPA		*p*-VALUE (MWW)	
	RECALL	PRECISION	RECALL	PRECISION	RECALL	PRECISION
(a)	0.6005 ± 0.0110	0.5375 ± 0.0057	0.5996 ± 0.0362	0.4709 ± 0.0150	0.4325	0.0000 ↑
(b)	0.6089 ± 0.0318	0.5322 ± 0.0194	0.6269 ± 0.0219	0.4579 ± 0.0118	0.0003 ↑	0.0000 ↑
(c)	0.6471 ± 0.0171	0.5093 ± 0.0097	0.6544 ± 0.0218	0.4428 ± 0.0114	0.0318 ↓	0.0000 ↑
(d)	0.6708 ± 0.0179	0.4970 ± 0.0102	0.6619 ± 0.0203	0.4387 ± 0.0112	0.0182 ↑	0.0000 ↑
(e)	0.5887 ± 0.0189	0.5434 ± 0.0106	0.6198 ± 0.0271	0.4605 ± 0.0132	0.0000 ↓	0.0000 ↑
(f)	0.591 ± 0.0218	0.5410 ± 0.0104	0.5570 ± 0.0393	0.4935 ± 0.0177	0.0000 ↑	0.0000 ↑
average	0.6180	0.5267	0.6199	0.4607	–	–

TABLE 5 Test times (mean ± standard deviation) of the GP edge detectors (in seconds).

TRAINING IMAGE	MGP	GPA
(a)	0.0569 ± 0.0327	$0.0335 \pm 0.0088 \uparrow$
(b)	0.0632 ± 0.0343	$0.0163 \pm 0.0155 \uparrow$
(c)	0.0713 ± 0.0323	$0.0334 \pm 0.0123 \uparrow$
(d)	0.0697 ± 0.0373	$0.0312 \pm 0.0074 \uparrow$
(e)	0.0655 ± 0.0353	$0.0331 \pm 0.0080 \uparrow$
(f)	0.0585 ± 0.0334	$0.0315 \pm 0.0097 \uparrow$

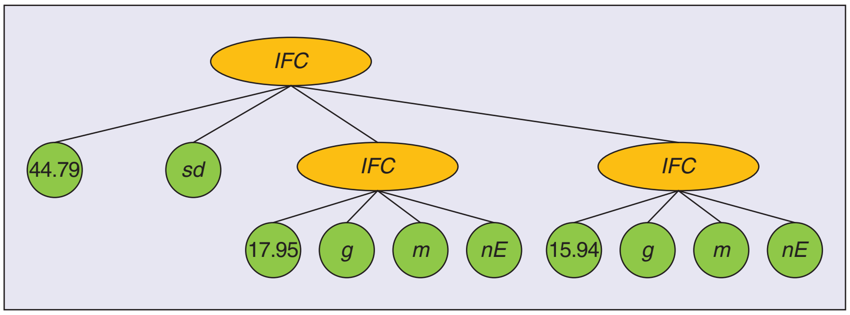

FIGURE 3 Example GP edge detector gp_{old} (with $F = 0.5304$) evolved by GPA [12].

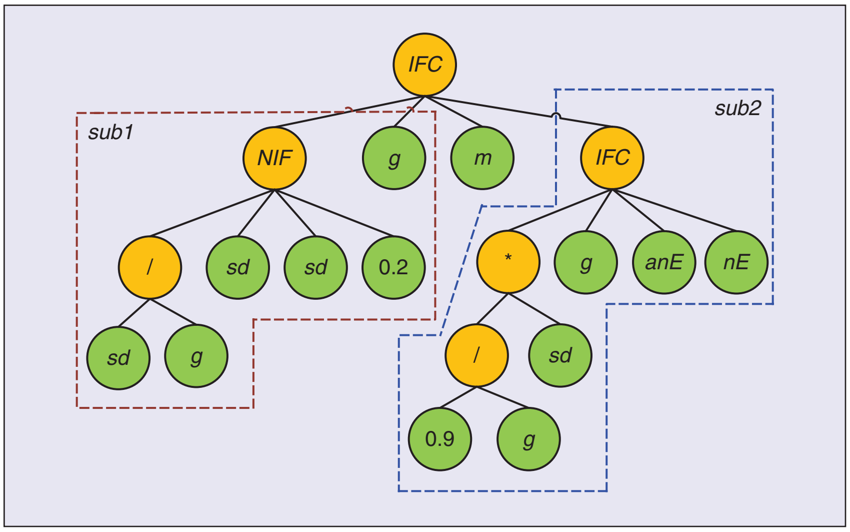

FIGURE 4 Example GP edge detector gp_{new} (with F = 0.5723) evolved by MGP.

high threshold is too low, some noisy pixels might be considered as edge points, and when finding weak edge points, more noisy pixels may be marked as edge points. Without ground truth, it is still an open issue to investigate how to effectively search for the best two thresholds [17, 49]. In this paper, we only use a single threshold in the Canny edge detector to quickly search for "good" binary edge maps. A fixed set of thresholds $(i/52, i = 1, 2, \ldots, 51)$ are given to obtain F values based on the ground truth of the 100 test BSD images. $F = 0.56$ is obtained as the maximum F from the 51 thresholds for the Canny edge detector, and it is the same as reported in [28].

Note that the Canny edge detector uses a set of thresholds and obtains different results. The test performance is the best test performance (maximum F) from all these results. The GP evolved edge detectors directly generate the final binary edge maps, and do not involve multiple thresholds to choose. The comparisons show that MGP has the ability to effectively evolve edge detectors when only a single image without ground truth is employed as the whole training set.

C. Computational Cost

As discussed in [12], GPA has a low computational cost. Table 5 provides the test times (in seconds) of the GP edge detectors evolved by MGP and GPA. All the experiments are based on a system with CPU 2.1 *GHz* and RAM 2*GB*. T-tests a significance level of 0.05 are used to compare their computational cost. "↑" means that GPA is significantly faster than MGP. It can be seen that the programs evolved by MGP have significantly longer test times than that of GPA, but the average test times in MGP are still much shorter than 0.1 second per image. Therefore, the MGP edge detectors' computational costs are still well within the 0.1 second requirement in real-time applications. Of course, a more powerful CPU can be used to further reduce the computational time on the test images.

D. Example GP Edge Detectors

Fig. 3 shows an edge detector gp_{old} (with a commonly found structure) evolved by GPA, while Fig. 4 gives an example of the edge detector gp_{new} evolved by MGP.

GPA focused mainly on the combinations of conditions in function *IFC*. If the condition in *IFC* is true, *IFC* calls the left sub-program; otherwise, it calls the right sub-program. From the structure of GP edge detector gp_{old}, the standard deviation *sd* in the root is helpful to choose a threshold on image gradient *g*. When *sd* is high, a threshold on *g* for images with noise needs to be larger than that of without noise.

For MGP, the image intensity in MGP is from 0 to 1. As can be seen from Fig. 4, the edge detector gp_{new} includes mainly two sub-parts *sub*1 and *sub*2. *sub*1 works mainly on marking edge points and *sub*2 works on how to mark pixels as non-edge points. There are three interesting observations.

Firstly, sub-part *sub*1 is constructed as a condition to discriminate pixels as edge points or non-edge points. Rather than using a fixed threshold on the image gradient *g*, the automatically constructed *sub*1 includes a condition using *sd* in addition to a fixed threshold. As a result, for an edge pixel from a noisy area, its *sd* and *g* are not low, so a fixed single threshold is chosen by *sub*1, and this threshold is expected to identify this pixel as an edge point. If the fixed threshold with a high value is used to consider the pixel as a non-edge point, some pixels (true edge points) in non-noisy areas are considered as non-edge points because their gradients are not large. Sub-part *sub*1 adaptively employs a fixed threshold, *sd* and *g* to mark pixels as edge points or non-edge points for noisy areas and non-noisy areas.

Secondly, sub-part *sub*2 is utilized as a condition to mark a single pixel as a non-edge point or a set of pixels as non-edge points. When a pixel has a very low *g* and a low *sd*, its neighbors are usually located in a non-edge area. *sub*2 can quickly mark its neighbors as non-edge points. However, when its *g* and *sd* are not low, it is hard to determine whether its neighbors are located in a non-edge area. From the evolved condition in *sub*2, the relationship between *g* and *sd* influences the ability of identifying a set of pixels being located in an edge area or a non-edge area.

Thirdly, similar to the detector gp_{old}, the detector gp_{new} utilizes the combination of *sd* and *g* to mark pixels as edge points or non-edge points. Although we do not pre-define condition functions in MGP, the detector gp_{new} has evolved adaptive conditions to mark pixels.

From the example detector gp_{new}, we can easily see that MGP can effectively evolve adaptive conditions for extracting edges, not directly using a single fixed threshold.

E. Visual Results

Fig. 5 shows examples of the detected results from the edge detectors evolved by GPA using each of the six training images (a)–(f) as the training set. It is

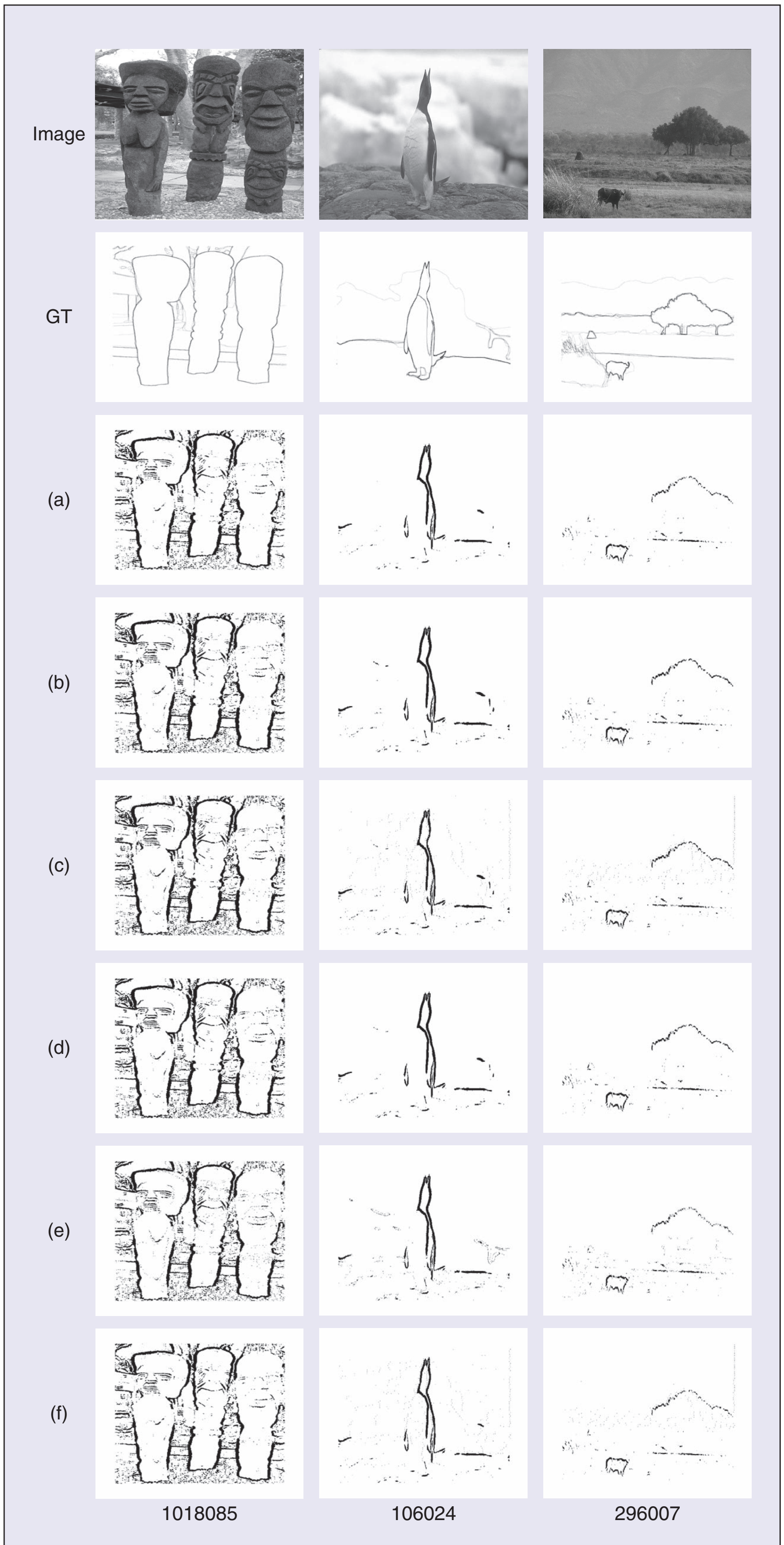

FIGURE 5 Examples of the detected results on three example BSD images from GPA [12] ("GT" is ground truth). The grayscale for GT and the detected results are inverted.

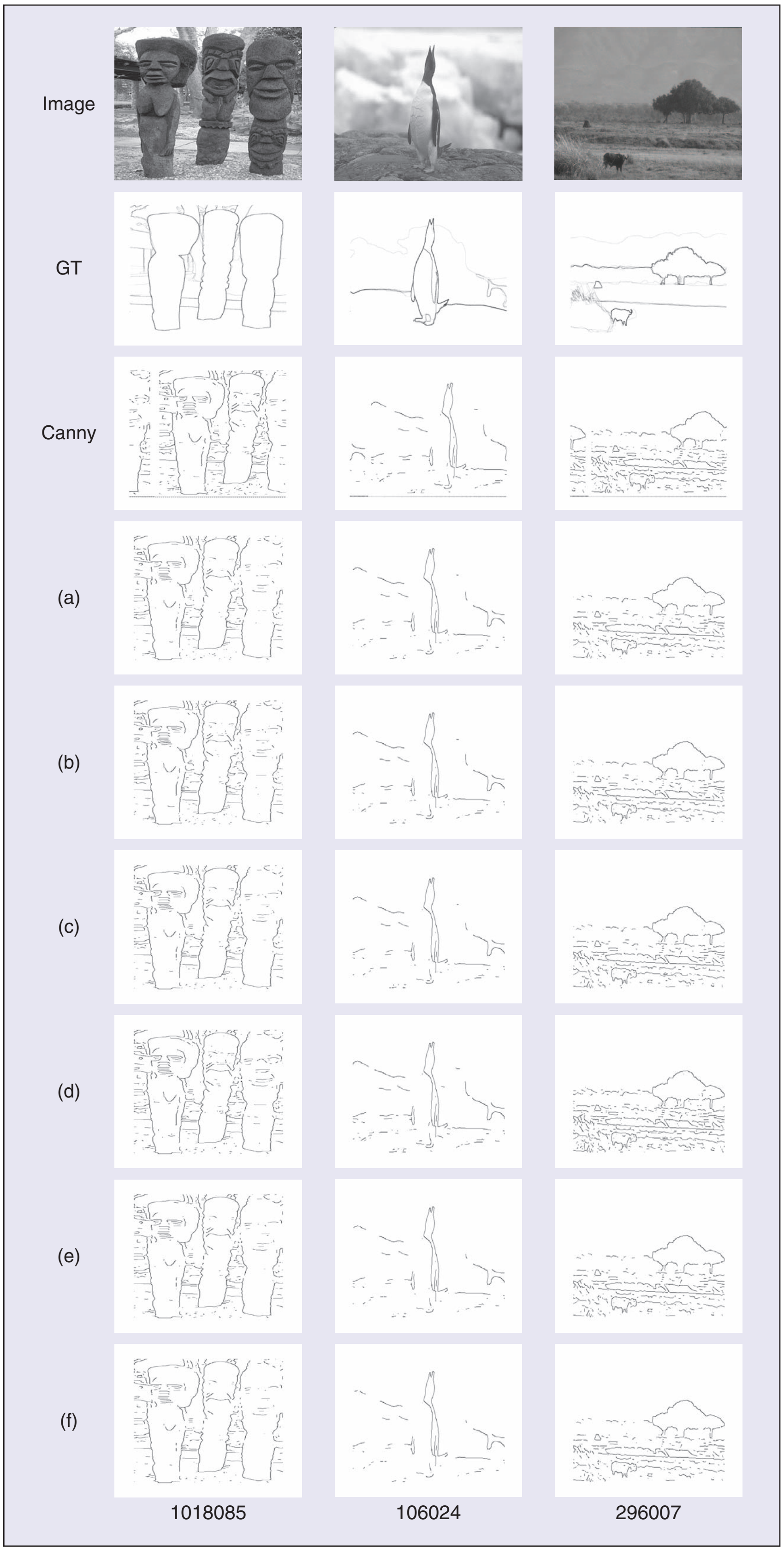

FIGURE 6 Examples of the detected results on three example BSD images from the GP edge detectors evolved by MGP ("GT" is ground truth). The grayscale for GT and the detected results are inverted.

found that the detected results from the six training images are quite similar. Without providing ground truth, GPA can effectively evolve edge detectors based on the image gradient.

Fig. 6 shows examples of the detected results from the Canny edge detector and the edge detectors evolved by MGP. Comparing with the Canny edge detector (i.e. the results with best *F* based on the ground truth), the GP edge detectors evolved by MGP achieve higher precision. For instance, for image 101085, the Canny edge detector has more false alarms than the edge detectors evolved by MGP. The MGP evolved edge detectors is significantly better than the Canny edge detector, more than 1% increase in terms of the F-measure. This is probably because the Canny edge detector uses only a fixed threshold for all the test images, but MGP has the ability to automatically construct conditions adaptively using the image gradient, the local standard deviation, and a fixed threshold.

Comparing the detected results in Figs. 5 and 6, MGP evolves edge detectors with single edge responses. A reason for this is that the energy (the image gradient) used in MGP is further processed by non-maximum suppression. These results suggest that MGP also has the ability to evolve edge detectors with similar detection performance when using (different) single training images without ground truth.

F. Single Training Image

This subsection discusses why a single image can be used by MGP and GPA as the whole training set to evolve edge detectors with good detection performance. This is also to continue the investigation on how to effectively evolve edge detectors using a single image, done initially in our previous work in the supervised learning scenario in [8].

For both supervised learning (with ground truth) and unsupervised learning (without ground truth), pre-defined specific knowledge, considered as prior domain knowledge, is helpful to find edge detectors when only a single image is used as the training data. In GPA, the image gradient and the standard deviation are considered as the prior knowledge. However, unlike existing thresholding techniques [16, 17], GP automatically evolves programs for marking edge points, rather than directly using one or two thresholds. These evolved programs can be considered as adaptive methods, which can effectively employ the image gradient and the standard deviation on different areas of the image, such as the examples shown in Figs. 3 and 4. To find such adaptive rules, the single training image should include various edge and non-edge information, such as texture, noise, and differentiation between two objects. This is probably why MGP can effectively evolve edge detectors when only a single image is used.

One-shot learning has been applied to object recognition, using a small set of training images [7, 50]. One-shot learning employs a very small set of training examples to train new classifiers, but needs prior knowledge extracted from existing datasets, such as learnt classifiers or pre-defined feature distributions, and such information is often obtained from a large dataset. Different from one-shot learning, MGP in this paper only relies on the pre-defined features (the image gradient and the standard deviation) and the given single image without any prior knowledge. Thus, MGP can be easier and faster to apply to train edge detectors than one-shot learning.

Note that in this work an image is used as the training data only if it has rich edge information. If an image has little edge information, such as cases with no or minor changes in a single color object, the edge detector evolved by MGP might not work well on other images. We will investigate how to determine whether the edge information of a single image is rich enough in the future.

VI. Conclusions

The goal of this paper was to develop an unsupervised learning GP system to improve the edge detection precision without adversely affecting the recall while keeping the low computational cost. The goal has been achieved by designing a terminal set, a function set, and a fitness function to propose a new algorithm named MGP. MGP uses only a single image as the whole training set to automatically evolve/generate effective edge detectors. Six images without ground truth were individually used by MGP for evolving effective edge detectors in the experiments. The results show that MGP can evolve edge detectors to obtain significantly better detection performance than the baseline algorithm GPA and the best performance of the Canny edge detector. The superior performance of MGP over GPA is probably due to the automatic condition rule construction and integrating non-maximum suppression, which also addresses the edge thickness problem. Further analysis on the evolved edge detectors reveals that MGP automatically combined the image gradient and the standard deviation with a threshold to evolve adaptive edge detectors, rather than using a single threshold only, to mark pixels as edge points or non-edge points.

This paper focuses mainly on the automatic construction of edge detectors using a single image, which is not specific to any particular domains, such as medical images. In the future, we will investigate edge detection and image analysis for medical images. Furthermore, we also aims to reduce the computational cost of MGP, and design a new marker as a terminal in GP to mark all points in certain local areas that include non-edge points only.

Acknowledgments

This work was supported in part by the Marsden Fund of New Zealand Government under Contracts VUW1209, VUW1509 and VUW1615, Huawei Industry Fund E2880/3663, and the University Research Fund at Victoria University of Wellington 209862/3580, and 213150/3662.

References

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

[2] M. Basu, "Gaussian-based edge-detection methods: A survey," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.(1995–2012)*, vol. 32, no. 3, pp. 252–260, Dec. 2002.

[3] C. I. Gonzalez, P. Melin, J. R. Castro, O. Castillo, and O. Mendoza, "Optimization of interval type-2 fuzzy systems for image edge detection," *Appl. Soft Comput.*, vol. 47, pp. 631–643, Oct. 2016.

[4] R. Medina-Carnicer and F. Madrid-Cuevas, "Unimodal thresholding for edge detection," *Pattern Recog.*, vol. 41, no. 7, pp. 2337–2346, July 2008.

[5] R. Medina-Carnicer, F. Madrid-Cuevas, A. Carmona-Poyato, and R. Muñoz-Salinas, "On candidates selection for hysteresis thresholds in edge detection," *Pattern Recog.*, vol. 42, no. 7, pp. 1284–1296, July 2009.

[6] K. Ray, "Unsupervised edge detection and noise detection from a single image," *Pattern Recog.*, vol. 46, no. 8, pp. 2067–2077, Aug. 2013.

[7] F.-F. Li, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 594–611, Apr. 2006.

[8] W. Fu, M. Johnston, and M. Zhang, "Is a single image sufficient for evolving edge features by genetic programming?" in *Proc. Int. Conf. Applications Evolutionary Computation: EvoApplications*, 2014, pp. 451–463.

[9] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[10] C. Harris and B. Buxton, "Evolving edge detectors with genetic programming," in *Proc. 1st Annu. Conf. Genetic Programming*, 1996, pp. 309–314.

[11] R. Poli, "Genetic programming for image analysis," in *Proc. 1st Annu. Conf. Genetic Programming*, 1996, pp. 363–368.

[12] W. Fu, M. Johnston, and M. Zhang, "Unsupervised learning for edge detection using genetic programming," in *Proc. IEEE Congr. Evolutionary Computation*, 2014, pp. 117–124.

[13] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, Nov. 1986.

[14] W. Fu, M. Johnston, and M. Zhang, "Genetic programming for edge detection: A Gaussian-based approach," *Soft Comput.*, vol. 20, no. 3, pp. 1231–1248, Mar. 2016.

[15] X. Lu, J. Yao, L. Li, Y. Liu, and W. Zhang, "Edge chain detection by applying Helmholtz principle on gradient magnitude map," in *Proc. 23rd Int. Conf. Pattern Recognition*, 2016, pp. 1364–1369.

[16] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *J. Electron. Imaging*, vol. 13, no. 1, pp. 146–168, Jan. 2004.

[17] P. Kaur and R. Maini, "Performance evaluation of various thresholding methods using Canny edge detector," *Int. J. Comput. Appl.*, vol. 71, no. 9, pp. 26–32, Dec. 2013.

[18] W. Fu, M. Johnston, and M. Zhang, "Genetic programming for edge detection: A global approach," in *Proc. IEEE Congr. Evolutionary Computation*, 2011, pp. 254–261.

[19] J. T. Fulton, Processes in biological vision online Corona Del Mar CA. USA vision concepts, 2004. [Online]. Available: http://neuronresearch.net/vision/. Accessed on: Aug. 27, 2018.

[20] W. Fu, M. Johnston, and M. Zhang, "Soft edge maps from edge detectors evolved by genetic programming," in *Proc. IEEE Congr. Evolutionary Computation*, 2012, pp. 24–31.

[21] S. Wang, T. Kubota, J. M. Siskind, and J. Wang, "Salient closed boundary extraction with ratio contour," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, pp. 546–561, Apr. 2005.

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

[23] S. Raut, M. Raghuvanshi, R. Dharaskar, and A. Raut, "Image segmentation: A state-of-art survey for prediction," in *Proc. Int. Conf. Advanced Computer Control*, 2009, pp. 420–424.

[24] N. Otsu, "A threshold selection method from gray-level histogram," *IEEE Trans. Syst., Man, Cybern.* (1971–1995)*, vol. 9, no. 1, pp. 62–66, Jan. 1979.
[25] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, vol. 1, pp. 321–331, Jan. 1988.
[26] T. Chan and L. Vese, "Active contours without edges," *IEEE Trans. Image Process.*, vol. 10, no. 2, pp. 266–277, Feb. 2001.
[27] L. Ganesan and P. Bhattacharyya, "Edge detection in untextured and textured images: A common computational framework," *IEEE Trans. Syst. Man, Cybern. B, Cybern. (1995–2012)*, vol. 27, no. 5, pp. 823–834, Sept. 1997.
[28] D. Martin, C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 530–549, May 2004.
[29] D. J. Park, K. M. Nam, and R.-H. Park, "Edge detection in noisy images based on the co-occurrence matrix," *Pattern Recog.*, vol. 27, no. 6, pp. 765–775, June 1994.
[30] W. Fu, M. Johnston, and M. Zhang, "Genetic programming for automatic construction of variant features in edge detection," in *Proc. Int. Conf. Applications of Evolutionary Computation: EvoApplications*, 2013, pp. 354–364.
[31] W. Fu, M. Johnston, and M. Zhang, "Low-level feature extraction for edge detection using genetic programming," *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1459–1472, Aug. 2014.
[32] C. Lopez-Molina, B. De Baets, and H. Bustince, "Quantitative error measures for edge detection," *Pattern Recog.*, vol. 46, no. 4, pp. 1125–1139, Apr. 2013.
[33] M. Donoser, H. Riemenschneider, and H. Bischof, "Linked edges as stable region boundaries," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010, pp. 1665–1672.
[34] Y.-K. Huo, G. Wei, Y.-D. Zhang, and L.-N. Wu, "An adaptive threshold for the Canny operator of edge detection," in *Proc. Int. Conf. Image Analysis and Signal Processing*, 2010, pp. 371–374.
[35] W. Fu, M. Johnston, and M. Zhang, "Distribution-based invariant feature construction using genetic programming for edge detection," *Soft Comput.*, vol. 19, no. 8, pp. 2371–2389, Aug. 2015.
[36] N. Fernández-García, A. Carmona-Poyato, R. Medina-Carnicer, and F. Madrid-Cuevas, "Automatic generation of consensus ground truth for the comparison of edge detection techniques," *Image Vis. Comput.*, vol. 26, no. 4, pp. 496–511, Apr. 2008.
[37] Y. Zhang and P. I. Rockett, "Evolving optimal feature extraction using multi-objective genetic programming: A methodology and preliminary study on edge detection," in *Proc. Genetic and Evolutionary Computation Conf.*, 2005, pp. 795–802.
[38] T. Golonek, D. Grzechca, and J. Rutkowski, "Application of genetic programming to edge detector design," in *Proc. Int. Symp. Circuits and Systems*, 2006, pp. 4683–4686.
[39] E. Bolis, C. Zerbi, P. Collet, J. Louchet, and E. Lutton, "A GP artificial ant for image processing: Preliminary experiments with EASEA," in *Proc. 4th European Conf. Genetic Programming*, 2001, pp. 246–255.
[40] S. Harding and W. Banzhaf, "Genetic programming on GPUs for image processing," *Int. J. High Performance Syst. Architecture*, vol. 1, no. 4, pp. 231–240, Mar. 2008.
[41] M. Ebner, "On the edge detectors for robot vision using genetic programming," in *Proc. Horst-Michael Groβ, Workshop SOAVE 97 – Selbstorganisation von Adaptivem Verhalten*, 1997, pp. 127–134.
[42] M. I. Quintana, R. Poli, and E. Claridge, "Morphological algorithm design for binary images using genetic programming," *Genetic Program. Evolvable Mach.*, vol. 7, pp. 81–102, Mar. 2006.
[43] J. Wang and Y. Tan, "A novel genetic programming based morphological image analysis algorithm," in *Proc. 12th Annu. Conf. Genetic and Evolutionary Computation*, 2010, pp. 979–980.
[44] I. Kadar, O. Ben-Shahar, and M. Sipper, "Evolution of a local boundary detector for natural images via genetic programming and texture cues," in *Proc. 11th Annu. Conf. Genetic and Evolutionary Computation*, 2009, pp. 1887–1888.
[45] W. Fu, M. Johnston, and M. Zhang, "Automatic construction of Gaussian-based edge detectors using genetic programming," in *Proc. 16th European Conf. Applications Evolutionary Computation*, 2013, pp. 365–375.
[46] U. Bhowan, M. Johnston, and M. Zhang, "Developing new fitness functions in genetic programming for classification with unbalanced data," *IEEE Trans. Syst. Man, Cybern. B, Cybern. (1995–2012)*, vol. 42, no. 2, pp. 406–421, 2012.
[47] L. Lam, S.-W. Lee, and C. Suen, "Thinning methodologies: A comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 9, pp. 869–885, Sept. 1992.
[48] W. H. Kruskal, "Historical notes on the Wilcoxon unpaired two-sample test," *J. Amer. Stat. Assoc.*, vol. 52, no. 279, pp. 356–360, Sept. 1957.
[49] Y. Li, M. Paluri, J. M. Rehg, and P. Dollár, "Unsupervised learning of edges," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 1619–1627.
[50] H. Al-Sahaf, M. Zhang, and M. Johnston, "A one-shot learning approach to image classification using genetic programming," in *Proc. 26th Australasian Joint Conf. Artificial Intelligence*, 2013, vol. 8272, pp. 110–122.

President's Message

(continued from page 4)

caused, whether it is congenital, or acquired in childhood or adulthood, is likely to play important roles and influence the performance of the system. Note that a BCI game is more important for a person with missing upper limbs. The situation becomes more challenging if we consider the fact that a particular area of the brain is no more in charge of a body part but in charge of activities that are done by that body part. Thus if we want to design a BCI based system, as an example, for people with missing upper limbs, it appears that we should use data from subjects with missing limbs. But this certainly poses a challenge to generate adequate data for designing such a system. This raises other important questions: To design a machine learning system for motor imagery, can we use imagination of lip movement or movement of the feet to do the same task? Using a BCI system how can we analyze what a person is actually doing (not functionally, but physically)? I have no answer to all these questions but they all appear to be challenging and are likely to impact designing of AI systems based on BCI.

References

[1] D. C. Knill and A. Pouget, "The Bayesian brain: The role of uncertainty in neural coding and computation," *Trends Neurosci.*, vol. 27, no. 12, pp. 712–719, 2004.
[2] K. Friston, J. Kilner, and L. Harrison, "A free energy principle for the brain," *J. Physiology-Paris*, vol. 100, no. 1–3, pp. 70–87, 2006.
[3] K. Friston, "The free-energy principle: A unified brain theory?" *Nature Rev. Neurosci.*, vol. 11, no. 2, pp. 127–138, 2010.
[4] N. Birbaumer, W. Lutzenberger, P. Montoya, W. Larbig, K. Unertl, S. Töpfner, W. Grodd, E. Taub, and H. Flor, "Effects of regional anesthesia on phantom limb pain are mirrored in changes in cortical reorganization," *J. Neurosci.*, vol. 17, no. 14, pp. 5503–5508, 1997.
[5] R. Chen, B. Corwell, Z. Yaseen, M. Hallett, and L. G. Cohen, "Mechanisms of cortical reorganization in lower-limb amputees," *J. Neurosci.*, vol. 18, no. 9, pp. 3443–3450, 1998.
[6] P. Montoya, K. Ritter, E. Huse, W. Larbig, C. Braun, S. Töpfner, W. Lutzenberger, W. Grodd, H. Flor, and N. Birbaumer, "The cortical somatotopic map and phantom phenomena in subjects with congenital limb atrophy and traumatic amputees with phantom limb pain," *Eur. J. Neurosci.*, vol. 10, no. 3, pp. 1095–1102, 1998.
[7] H. Flor, T. Elbert, W. Mühlnickel, C. Pantev, C. Wienbruch, and E. Taub, "Cortical reorganization and phantom phenomena in congenital and traumatic upper-extremity amputees," *Exp. Brain Res.*, vol. 119, no. 2, pp. 205–212, 1998.
[8] W.-H. Qiu, H.-X. Wu, Q.-L. Yang, Z. Kang, Z.-C. Chen, K. Li, G.-R. Qiu, C.-Q. Xie, G.-F. Wan, and S.-Q. Chen, "Evidence of cortical reorganization of language networks after stroke with subacute Broca's aphasia: A blood oxygenation level dependent-functional magnetic resonance imaging study," *Neural Regener. Res.*, vol. 12, no. 1, pp. 109–117, 2017.
[9] A. Hahamy, S. N. Macdonald, F. van den Heiligenberg, P. Kieliba, U. Emir, R. Malach, H. Johansen-Berg, P. Brugger, J. C. Culham, and T. R. Makin, "Representation of multiple body parts in the missing-hand territory of congenital one-handers," *Current Biol.*, vol. 27, no. 9, pp. 1350–1355, 2017.