Genetic Programming For Edge Detection: A Global Approach

Wenlong Fu, Mark Johnston
School of Mathematics, Statistics and Operations Research
Victoria University of Wellington, New Zealand
Email: {wenlong.fu,mark.johnston}@msor.vuw.ac.nz

Mengjie Zhang
School of Engineering and Computer Science
Victoria University of Wellington, New Zealand
Email: mengjie.zhang@msor.vuw.ac.nz

***Abstract*—Edge detection is an important task in computer vision. This paper describes a global approach to edge detection using genetic programming (GP). Unlike most traditional edge detection methods which use local window filters, this approach directly uses an entire image as input and classifies pixels directly as edges or non-edges without preprocessing or postprocessing. Shifting operations and common standard operators are used to form the function set. Precision, recall and true negative rate are used to construct the fitness functions. This approach is examined and compared with the Laplacian and Sobel edge detectors on three sets of images providing edge detection problems of varying difficulty. The results suggest that the detectors evolved by GP outperform the Laplacian detector and compete with the Sobel detector in most cases.**

I. Introduction

Edge detection is a well developed area of image analysis. Many different techniques have been developed based on window filters [1]. For example, the gradient-based methods search the maximum and minimum in the first derivative of the images. Laplacian-based methods find the zero crossings in the second derivative of the images [2]. Some model-based methods typically design a single model for determining pixels' labels [3][4]. Most of these techniques are designed based on local information and changes from pixels' intensities, which often weaken the global information and are typically sensitive to noise. Some of these edges detectors highlight local discontinuities among different regions, which may lead to some useful features missing.

Genetic programming (GP) has been employed for object detection and image analysis since the 1990s [5][6][7][8] and achieved great success. However, there are only a few reports for edge detection. Most GP approaches to edge detection follows the conventional way by using a small window with specific features as the input and use image processing operators to form the function set. The investigation of whether GP can evolve detectors without domain specific knowledge and local information has not been intensively explored.

A. Goals

This paper aims to investigate a domain independent, global approach to the use of GP for edge detection without using local information of the images. Instead of using a local window in a moving fashion to detect edges in entire images as in many existing methods, this approach will directly using the entire/full images as the inputs to the GP system. In addition, this approach will also avoid using predefined (specific) image features. This approach will be examined and compared with two common edge detectors (Laplacian and Sobel) on three sets of images providing edge detection problems of varying difficulty. Specially, we investigate the following objectives:

- how the terminal set and function set can be formed;
- how the fitness function can be constructed;
- whether this approach can achieve good enough performance for edge detection;
- whether the detectors evolved by GP can compete with the Laplacian and Sobel detectors;
- whether the introduction of true negative measure into the F-measure based fitness function can improve edge detection performance; and
- whether the behaviour of the detectors evolved by GP can be interpreted.

Without losing generality, we will use grey-level images in our experiments. We will not perform any preprocessing or postprocessing(such as thresholding or other noise removal techniques) on the images, and will use the raw images directly for edge detection [9].

B. Organisation

In the remainder of the paper, Section II briefly describes the background of edge detection and genetic programming for edge detection. Section III develops a new method for edge detection. After presenting the experimental design in Section IV, Section V discusses the experimental results. Section VI gives conclusions and future work directions.

II. Background

This section briefly describes the background information for edge detection, and GP work related to edge detection.

A. Edge Detection

Edge detection is one of the most essential tasks in image processing and computer vision. For example, it is very useful for feature detection and extraction. The purpose of edge detection is to identify points in an image at which the pixel intensities change sharply or irregularly. Edge detection in untextured images aims at finding these interesting points based on local discontinuities. Edge detection in textured

978-1-4244-7835-4/11/$26.00 ©2011 IEEE

images is more complex, where these points are covered from the regular changes to the irregular changes.

Almost all edge detection methods are based on discontinuity, e.g., derivative-based methods such as the Sobel detector and the Laplacian detector [1]. Sobel detectors are based on the first derivative and Laplacian detectors on the zero crossings in the second derivative [10]. Based on the derivatives, smoothing techniques for reducing noise are introduced [2]. Gaussian filters are also widely used in edge detection. From approximation to the shape of spatial receptive fields, Gaussian filters along with the Laplacian detectors are very similar to the difference of Gaussians (DOG) [2]. Canny detectors [11] are derived from an optimal edge detector which turns out to be well approximated by the derivative of a Gaussian function [12]. The popular strategy is to design a local window to filter non-edge points with a threshold. The window size is usually 3×3.

Besides derivative methods, some other methods consider the window filter as a model and then search the model. For instance, Terry and Vu trained a 3×3 edge detection filter by Neural Networks [13]. The common property in almost all approaches based on one window is that different directions of the edges exist in the window so that they can find the hypothetical edges. Regional methods and global methods have also been applied to edge detection, and linear invariant filtering is an example of this kind [14]. A common characteristic of these approaches is that the prior local information of the images (edges) is required to design the filter.

B. Related Work to Genetic Programming For Edge Detection

GP [15] has been applied to various fields [16][17][18][19]. GP can create programs or functions for a task based on "what to do", which is different from other methods based on "how to do it". GP can evolve solutions without knowing the details of "how to do it" and these solutions are often human competitive [19]. However, there is little previous work using GP for edge detection. Harris and Buxton [20] designed approximate response detectors in one-dimensional signals by GP, but this is based on theoretical analysis of an ideal edge detector and its corresponding properties. Ebner [21] used four shift functions (similar to the macros in Poli [22]) and other functions to approximate the Canny edge detector. Bolis et al. [23] simulated an artificial ant to search for edges in an image. Zhang and Rockett [3] evolved a 13×13 window filter for comparison with the Canny edge detector. Wang and Tan [4] used linear GP to find binary image edges, inspired by morphological operators, erosion and dilation, as terminals [24] for binary images. To date, there have been very few reports using GP for edge detection without using window filters or related image operators.

III. Methods

In this approach, standard tree-based GP [15] is used to construct the edge detectors. In the rest of this section, we will describe the terminal set, the function set and the fitness functions.

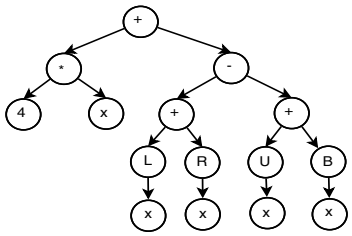


Fig. 1. Laplacian filter constructed with genetic programming. The nodes 'L', 'R', 'U' and 'B' represent the functions "$shift_{-1,0}$", "$shift_{1,0}$", "$shift_{0,-1}$" and "$shift_{0,1}$".)

A. Terminals

As a global method, we use the entire image as input to the GP system. So the terminal set is designed to contain the entire image x and random constants rnd in the range of $[-10, 10]$. As we are processing 8-bit grey-level images (PGM), x is a matrix with a maximum value of 255 and a minimum value of 0.

B. Functions

As we are using the entire image (rather than a local window information or features) as input, we will need shifting (translation) operators. Ebner [21] used four different shifting functions for the four directions (shift down, shift up, shift left and shift right). To make GP more effective at finding edges by automatically choosing the correct operations, we merge the four different shifting functions into one shifting function $shift_{n,m}$ so that genetic programming can easily find far neighbours. Here, function "$shift_{n,m}$" is a main operator, meaning that the image will shift by n columns and m rows. If n is negative, it shifts to left, otherwise shifts to right. If m is negative, it means shifting up, otherwise down. For example, "$shift_{1,0}$" and "$shift_{-1,0}$" mean that the input image will be shifted right and left by one pixel respectively; "$shift_{0,-1}$" and "$shift_{0,1}$" will make the input image shift up and down by one pixel respectively; and "$shift_{-1,-1}$" encapsulates the function shift left one pixel and shift up one pixel. In this way, we need fewer nodes in our program than that in [21].

We also include several commonly used operators such as the addition, subtraction, multiplication, division, square, square root and absolute value. So the function set is

$$\{+, -, \times, /, shift_{n,m}, abs, sqrt, square\}$$

All these functions are able to operate on the input image matrix. The $+, -, \times$, $abs, square$ have their usual meanings. The square root function "$sqrt$" is protected, which produces a result of 0 for negative inputs. Division "/" is also protected, producing a result of 1 for a 0 divisor.

Using this function set, a classical filter can be expressed as an individual program. For example, the Laplacian filter can be shown as "$(+ \quad (\times \quad 4.0 \quad x)\,(- \quad (+ \quad (shift_{0,1} \quad x)\,(shift_{0,-1} \quad x))\,(+ \quad (shift_{1,0} \quad x)\,(shift_{-1,0} \quad x)))$". The tree representation can be seen in Fig 1.

C. Fitness Functions

In this approach, we treat edge detection as a binary classification problem. If a genetic program output corresponding to a pixel is positive, then that pixel is classified as an edge pixel; otherwise, it is a background pixel (non-edge).

However, this is very different from normal classification in that the number of edge pixels is usually much smaller than non-edge pixels. In other words, the edge pixels in an image usually belong to the *minority* class. In normal classification problems, the classification accuracy or error rate (the number of pixels that are correctly or incorrectly classified as edge pixels as a proportion of the total number of pixels in the image) is used as a performance measure. However, if the accuracy or error rate is used as the fitness function for edge detection, it will lead individual programs to acquire good fitness by not detecting any edges. So some measures need to be considered to balance the real edge pixels and background pixels.

Martin et al. [25] used the *F-measure* to train edge detectors. As it is a balanced measure and used frequently in edge detection and object detection, we also use it to construct our fitness function.

To use the F-measure, we need to calculate recall and precision. *Recall* (r) is the number of pixels on the edges correctly detected as a proportion of the total number of pixels on the edges (in the ground truth). *Precision* (p) is the number of pixels on the edges correctly detected as a proportion of the total number of pixels detected as edges (including those non-edge pixels that are incorrectly detected as edges). Based on precision and recall, the (first) fitness function based on F-measure is shown in (1), where α is a relative cost (weighting factor) between recall r and precision p to represent the relative importance of them ($0 < \alpha < 1$).

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

We transfer edge detection to a minimisation problem and set $\alpha = 0.5$. The fitness function is defined as F_m (shown in equation (2)), where N is the number of training images, and r_i and p_i are recall and precision for image i.

$$F_m = \frac{1}{N}\sum_{i=1}^{N}\left(1 - \frac{2r_i p_i}{r_i + p_i}\right) \tag{2}$$

In theory, high recall would correspond to more edge pixels being correctly detected, and high precision to fewer non-edge pixels to be incorrectly detected as edges. As the above fitness function treats precision and recall as equally important ($\alpha = 0.5$) and there are typically more non-edge pixels in an image, it might cause a large number of non-edge pixels to be incorrectly detected as edges, which could lead to low specificity. *Specificity* (s), or true negative rate, is the number of non-edge pixels correctly classified as a proportion of the total number of non-edge pixels. This is clearly what we wanted. While changing (decreasing in this case) the weight factor α can solve the problem in theory (at least improve the situation), we directly introduce the specificity into the fitness measure and construct a second fitness function, as shown in equation (3). Here, s_i is specificity for image i.

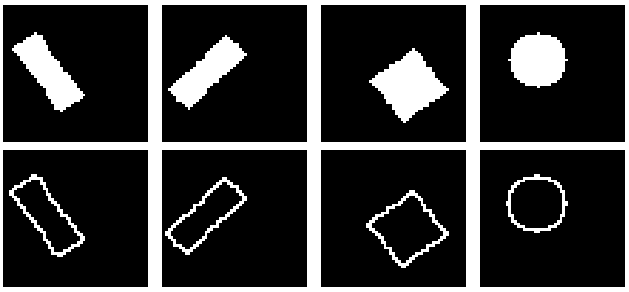


Fig. 2. Some of training binary images and the related ground truth

(a) t1 (b) t2 (c) t3 (d) t4

Fig. 3. Some of the test binary images

$$F_n = \frac{1}{N}\sum_{i=1}^{N}(1 - \sqrt{s_i r_i p_i}) \tag{3}$$

Note that the specificity is closely related to recall and precision (particularly precision) and is not an independent measure. We would like to investigate whether the direct introduction of the specificity measure into the fitness function will improve the edge detection performance.

IV. Experimental Design

This section describes the image datasets and parameter settings for our experiments.

A. Image Datasets

We used three image datasets in our experiments. These image sets provide edge detection problems of varying difficulty. The first set contains binary images designed for simple edge detection. The second image set consists of images chosen from reference [26] and the third set comes from Berkeley Segmentation Dataset (BSD) [27]. These second and third sets contain real images, where the edges are much harder to detect. When the image size becomes large, the training time will be long, and we can use sampling techniques to sample I_n small images (size $I_s \times I_s$) as inputs. For the images with simple edges in the binary images and for the four real images, we do not do use sampling. For the last set of images with complex edges, we do use sampling to train detectors.

1) Binary image dataset: We construct a binary image dataset consisting of 18 images with simple edges, nine of which are used for training and nine for testing. Fig 2 shows examples of training binary images with their corresponding ground truth images. Fig 3 shows some examples of the test images. There is only one shape in each image. These shapes have different orientations and are placed in different places

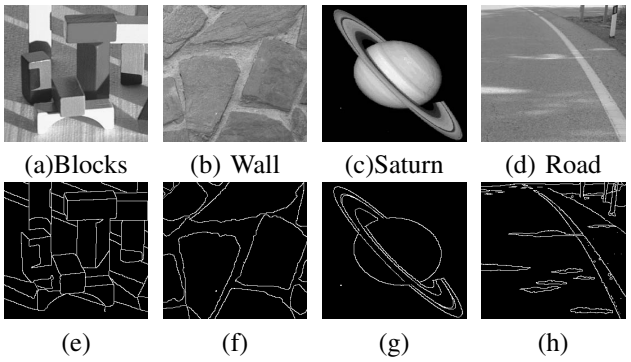


Fig. 4. Four real images and corresponding ground truth

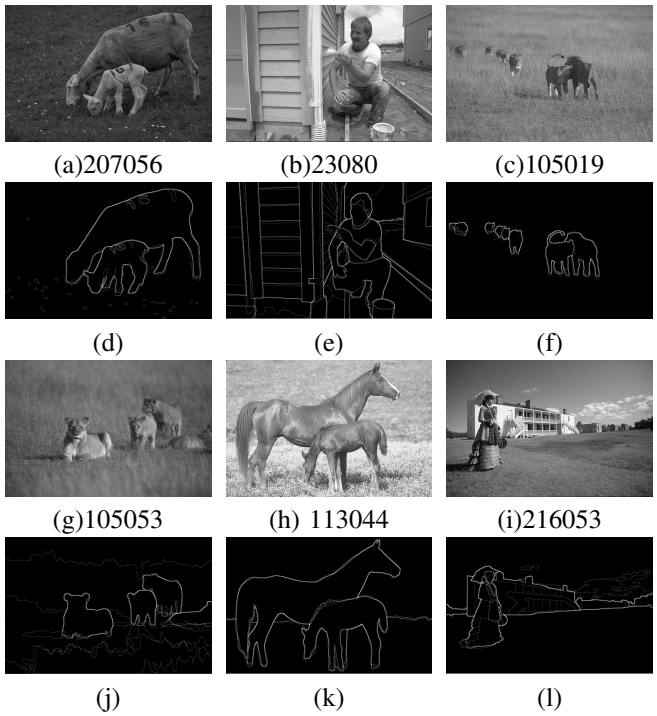


Fig. 5. Training images from BSD dataset and the related ground truth

in the images. For example, the rectangle images contain orientations $45°$, $145°$, $30°$ in training images and orientations $0°$, $90°$, $60°$ in test images.

2) Four real images: Fig 4 shows the four real images (a)–(d) and their corresponding ground truth images (e)–(h) taken from reference [26]. Compared with the simple edges in Fig 2, the edges in these images are much more complex, which is expected to be much harder to detect. Fig 4(a) and (b) are used as training images and Fig 4(c) and (d) are used as test images.

3) Images from BSD: The Berkeley Segmentation Dataset (BSD) [27] comes from natural images. Fig 5 shows six example training images and their corresponding ground truth. Ten images (Fig 6) are used as the test dataset. These images are very different. For example, images 253055 and 361010 (Fig 6(e) and (g)) have similar background with images 207056 and 216053 (Fig 5(a) and (i)). Image 175032 (Fig 6(b)) is more complicated than the training dataset because of the disorderly background. Compared with the images in the first and the second image sets, the edge detection problems in these images are clearly different.

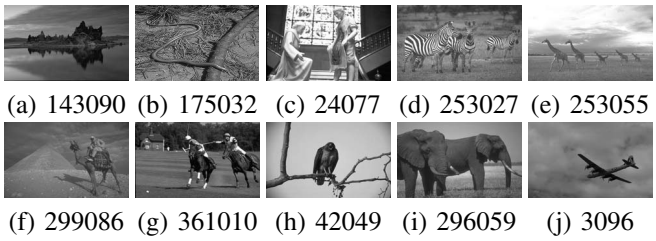


Fig. 6. Test images from BSD dataset

B. Parameter Setting

We use the standard one point crossover and one point mutation operator in our experiments. The crossover and mutation probabilities are $p_c = 0.80$, $p_m = 0.15$, respectively. The elitism rate is $p_e = 0.05$. The population size is 200 for all three image sets. The maximum number of generations is 250. In the first two image sets, we use the whole image as inputs. For the BSD image dataset, we sample 5 sub-images with size 41×41 for each image for training to reduce the computational cost. All constant numerical terminals ("rnd") are generated randomly from -10 to 10. These values are chosen based on common settings and initial experiments via empirical search. Each of the GP experiments is repeated for 30 independent runs using 30 random seeds and the average and the best results will be reported with significance test indications.

V. Experimental Results and Discussion

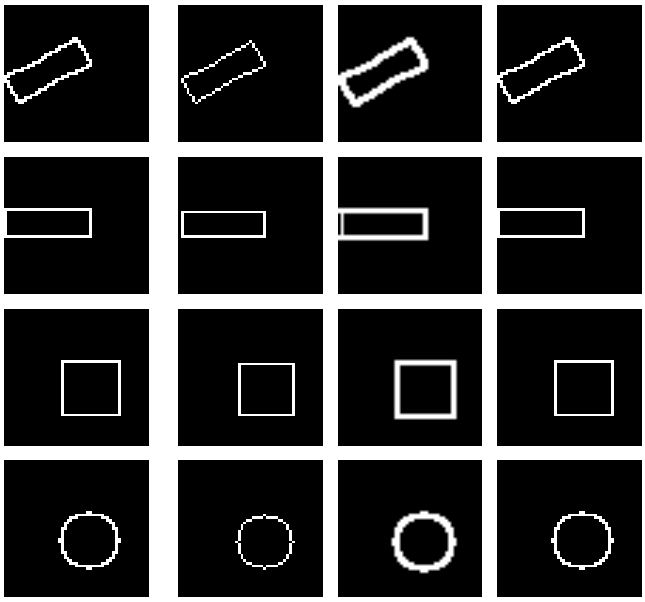
This section describes the experimental results of the GP approach and comparisons with the Sobel and Laplacian edge detectors. To make a fair comparison, we will use the same criterion for the three methods. For presentation convenience, we transfer the standard F-measure $\frac{2rp}{r+p}$ to $F = 1 - \frac{2rp}{r+p}$, in which case, the smaller the transferred F-measure, the better the performance. In addition, we also report the specificity (s) as a secondary criterion (false negative). In the rest of the section, we will describe the overall results, a comparison of the two fitness functions, and an analysis of an evolved genetic program detector.

A. Overall Results

1) Binary Image Dataset: Table I shows the results of the test images for GP(Best), GP(Average), Sobel and Laplacian detectors. In the table, images $t1$ to $t4$ come from Fig 3, and the "Average" column is the mean of all nine images in the test set. The first row "GP(Best)" gives the results (transferred F-measure value and the specificity) of the best evolved genetic program detector among all the 60 runs (30 runs using F_m and 30 runs using F_n) for each image. The second row "GP(Average)" refers to the results of the best genetic program detector in each run over all the 60 runs. The third and fourth rows give the results of the Sobel detector and the Laplacian detector. In addition, we also performed

TABLE I
COMPARISON BETWEEN GP, SOBEL AND LAPLACIAN DETECTORS FOR BINARY IMAGES

Image		t1	t2	t3	t4	Average
GP(Best)	F	0.0000	0.0000	0.0000	0.0000	0.0000
	s	1.0000	1.0000	1.0000	1.0000	1.0000
GP(Average)	F	0.0081	0.0049	0.0025	0.0051	0.0046
	s	1.0000	1.0000	1.0000	1.0000	1.0000
Sobel	F	0.3292^-	0.3793^-	0.3333^-	0.3226^-	0.3337
	s	0.9575^-	0.9683^-	0.9683^-	0.9682^-	0.9650
Laplacian	F	0.5517^-	0.5000^-	0.5000^-	0.5000^-	0.5045
	s	0.9891^-	0.9853^-	0.9853^-	0.9917^-	0.9884



(a) Ground Truth (b) Laplacian (c) Sobel (d) GP

Fig. 7. Edge detection for the binary images

statistical significance tests to compare the three methods by comparing GP with Sobel, and GP with Laplacian by a *one sample t-test* with significance level 0.05. Symbols "$-/+$" in the third row means that Sobel is statistically significantly worse/better than GP, using the numbers in the GP(Average) row and in the Sobel row. Note, there are no test for the column "Average".

According to Table I, GP performed significantly better than both Sobel and Laplacian on all the test images by considering the average results of GP over the 60 runs. Also, GP could achieve the ideal performance on all the test images (GP(Best) got a value of zero for the minimised F-measure and a value of 1.0 for specificity), meaning that all the actual edge pixels were correctly detected by GP without any false alarms. Inspection of the GP(Average) results reveals that over the 60 runs, most of them achieved an ideal result or very close to ideal (all average F values were less than 0.01). On the other hand, Sobel and Laplacian could not achieve such a level of results, although they are not bad either.

To give an intuitive view, Fig 7 shows some example best results for four test images achieved by the three methods together with their ground truth images. GP detected these images perfectly. The Laplacian detector has one pixel offset for those objects (some true edge pixels are missing). The Sobel detector has doubled most of the actual edge pixels for those objects. These results clearly show that GP outperformed Sobel and Laplacian on these images.

TABLE II
RESULTS FOR FOUR REAL IMAGES

Image		Blocks	Wall	Saturn	Road
GP(Best)	F	0.4378	0.6823	0.4290	0.5798
	s	0.9490	0.9731	0.9692	0.9770
GP(Average)	F	0.5113	0.7505	0.6452	0.6992
	s	0.9460	0.9610	0.7716	0.9727
Sobel	F	0.5614^-	0.7001^+	0.4151^+	0.8239^-
	s	0.9706	0.9386	0.9710	0.9966
Laplacian	F	0.8776^-	0.9372^-	0.7055^-	0.8014^-
	s	0.9455	0.8352	0.9906	0.9611

2) Four Real Images: Table II shows the results for the three methods on the four real images. Clearly, the results on this image set are worse than those on the binary image set for all the three detection methods, reflecting the fact that the edge detection problem in this image set is more difficult than that for the binary images with simple objects.

Another clear observation is that the s value (specificity) for the different methods on these images are very similar, which suggests that the measure of specificity is somehow redundant to the F-measure (precision and recall). This will be further investigated in the next subsection when we compare the two fitness functions.

Inspection of the transferred F-measure values of the different methods for these images reveals that the GP method significantly outperformed Laplacian for all the images. Although GP(Average) achieved very similar results to Sobel (significantly better on two images but also significantly worse on the other two images), the best program detectors evolved by GP are better than Sobel on most cases except for the Saturn image on which the F-measure values are very similar.

Fig 8 shows the example detected edges of the four images by the three methods. Clearly, GP outperformed Laplacian and Sobel on most cases, but GP produced more false alarms than Sobel on the Saturn image (although the best detector from GP is close to the Sobel detector). The main reason for that is the training set is too small (only two images) and does not cover the edge features of the type found in the Saturn image. In fact, considering this factor, GP achieved really good results.

3) BSD Dataset: Table III shows the results of the three methods for the ten images in the test set from the BSD image dataset. These results show a very similar pattern to the four real images set. Clearly, the best detectors evolved by GP (the first row) for all the 10 test images outperformed Sobel and Laplacian based on the primary measure of the transferred F-measure. Even for the average results over 60 runs, GP still achieved significantly better performance (transferred F-

TABLE III
AVERAGE RESULTS FOR TEN IMAGES FROM THE BSD DATASET

Image		143090	175032	24077	253027	253055	299086	361010	42049	296059	3096
GP (Best)	F	0.6284	0.9275	0.7859	0.8531	0.5737	0.7408	0.6838	0.3878	0.5172	0.3373
	s	0.9865	0.6850	0.8949	0.8521	0.9818	0.9827	0.9313	0.9812	0.9736	0.9945
GP (Average)	F	0.7291	0.9421	0.8166	0.8531	0.6655	0.7864	0.7368	0.5013	0.6282	0.5303
	s	0.9594	0.6322	0.8742	0.8535	0.9877	0.9714	0.9181	0.9680	0.9622	0.9905
Sobel	F	0.7471^{-}	0.9391^{+}	0.8421^{-}	0.9110^{-}	0.7359^{-}	0.8533^{-}	0.7598^{-}	0.4964	0.5930^{+}	0.4766^{+}
	s	0.9861	**0.000056**	0.9235	**0.0014**	0.9917	0.9863	0.9777	0.9847	0.9902	0.9943
Laplacian	F	0.8185^{-}	0.9469^{-}	0.9181^{-}	0.9140^{-}	0.8336^{-}	0.8921^{-}	0.8480^{-}	0.6963^{-}	0.8479^{-}	0.6399^{-}
	s	0.9684	0.5091	0.9252	0.9150	0.9702	0.9614	0.9724	0.9789	0.9488	0.9909

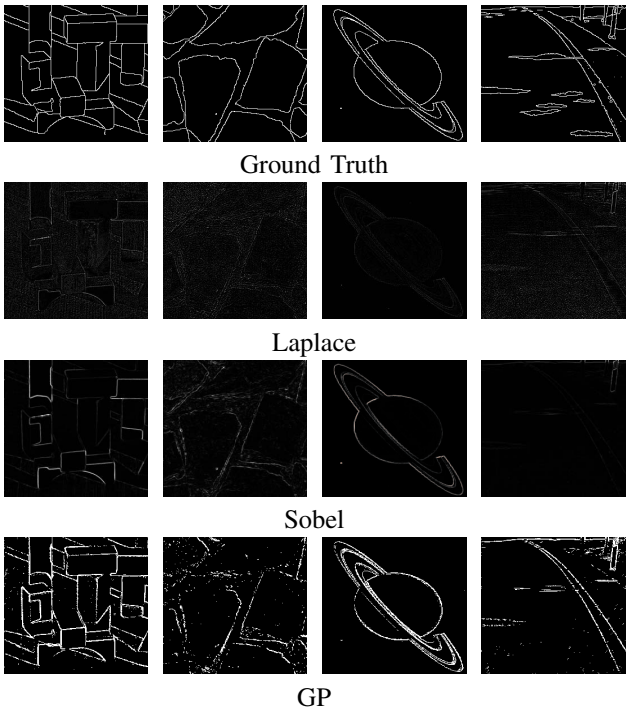


Fig. 8. Edge detection for the four real images

measure) than Laplacian for all the 10 images, and than Sobel for most images (six out of ten).

Regarding the specificity s, again, the values for different methods are very similar, indicating that this measure is redundant to the F-measure (precision and recall). The only exceptions are for images 175032 and 253027, where Sobel had very low values. This suggests that Sobel achieved high recall at a price of a huge number of false alarms (by treating a large number of non-edge pixels as edges). GP and Laplacian do not have this problem.

Fig 9 shows the ground truth and the detected images by the three methods on the 10 test images in the BSD image set. These detected image examples show that GP detected more details than the Sobel and Laplacian detectors. These results are very consistent with Table III.

Based on the overall results on the binary images with simple object edge, the real images and BSD images with complex edges, we can have the following observations.

- The proposed global GP approach can do a very good job for detecting for simply and complex edges on the (relatively clean) real/natural images.
- The detectors evolved by GP are very competitive with the Sobel and Laplacian edge detectors and outperform them in most images investigated here.
- The measure of specificity (true negative rate) seems redundant to the F-measure. In most cases, the specificity results are very similar for all the three methods. The next subsection will make further investigations on this point.

TABLE IV
COMPARISON BETWEEN F_m AND F_n FOR BINARY IMAGES

Image		t1	t2	t3	t4	Average
F_m	F	0.0125	0.0004	0.0004	0.0061	0.0046
	s	1.0000	1.0000	1.0000	1.0000	1.0000
F_n	F	0.0038	0.0094	0.0047	0.0041	0.0046
	s	1.0000	0.9999	1.0000	1.0000	1.0000

B. Comparison of the Two Fitness Functions

In Section III, we introduced specificity (true negative rate) directly into the fitness measure to constructed a second fitness function based on precision, recall and specificity (F_n). We would like to compare it to the first fitness function F_m to reveal whether F_n can improve the edge detection performance over F_m. The results of the previous subsection suggest that the specificity is very similar for most cases, indicating it might be redundant to the F-measure particularly the precision. So this subsection examines the details by directly comparing the two fitness functions in the global GP approach on the three image sets. Again, to make a fair comparison, we used the transferred minimisation F-measure and specificity as the evaluation criteria.

Table IV shows the comparison between fitness function F_m and F_n for the binary images containing simple objects. The average results over 30 independent runs for each fitness function are presented here. We also did a t-test with significant level 0.05 for all the images in this set. As can seen from this table, the results for both F and s are very similar and the corresponding values for the two fitness function are not statistically significantly different for any image in this dataset.

Table V shows the comparison between fitness function F_m and F_n for the four real images. While the evolved detectors

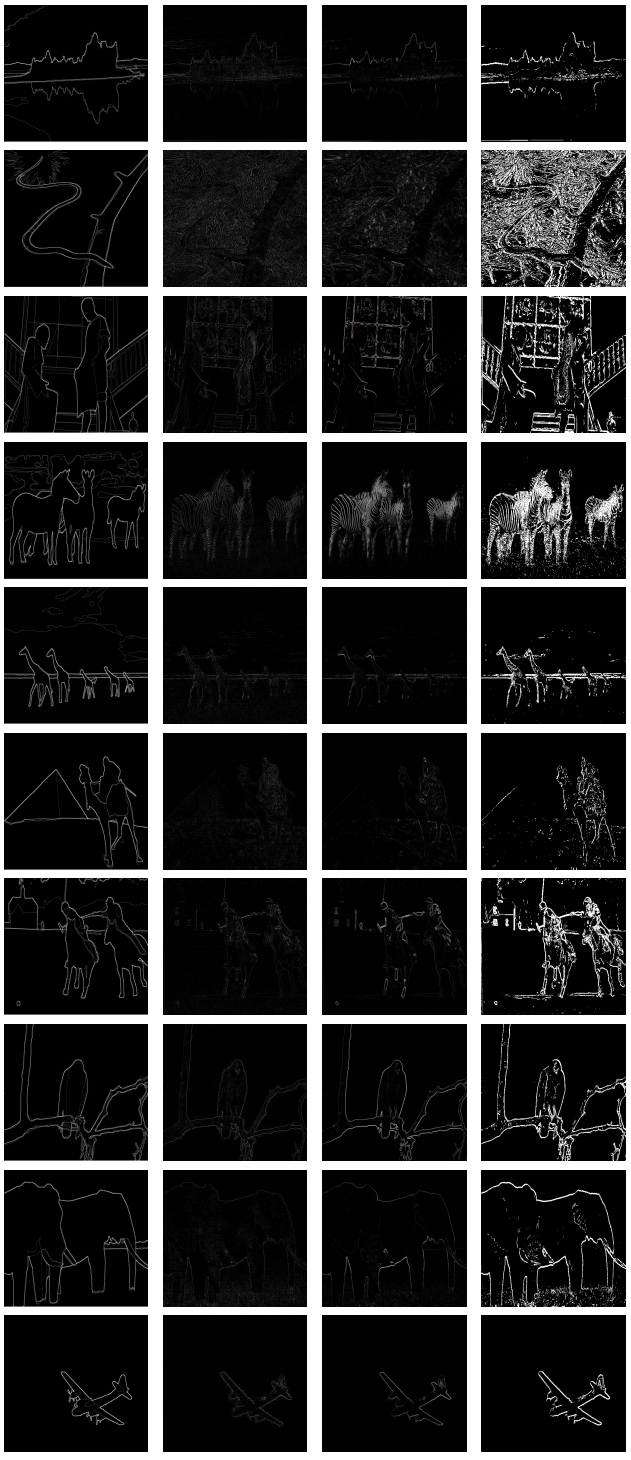


Fig. 9. Edge detection for BSD images

by GP with F_n are significantly better for the Blocks image than F_m, it is exactly the other way around for the Wall image. For the other (test) images, there is no significant difference between the two fitness functions. These results further indicate that the direct introduction of specificity into the F-measure based fitness function does not have significant effect.

TABLE V
COMPARISON BETWEEN F_m AND F_n FOR FOUR REAL IMAGES

		Blocks	Wall	Saturn	Road
F_m	F	0.5276	0.7253	0.6076	0.6967
	s	0.9436	0.9625	0.8272	0.9759
F_n	F	0.4951^+	0.7757^-	0.6828	0.7017
	s	0.9486	0.9594	0.7160	0.9695

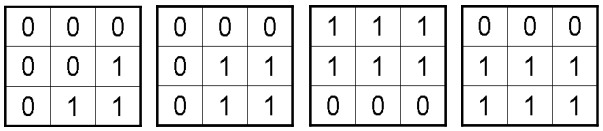


Fig. 10. Pattern for edge points or non-edge points

Table VI shows the comparison between detectors evolved by using fitness function F_m and F_n for the BSD images with complex object edges. The average results of 30 independent runs for each fitness function are reported. These results show a very similar pattern to those on the previous two data sets: the two fitness functions do not have significant difference in the results in most cases.

These experimental results on the three image sets confirm the theoretical analysis in Section III: the specificity measure is not really necessary when using fitness functions based on F-measure. This is mainly because it is not entirely independent of F-measure. The value of specificity can be derived from the precision, recall and the basic image size information.

C. Analysis of Evolved Edge Detectors

To investigate why the GP approach can evolve good edge detectors, we use the example program detector in (4) evolved based on the binary image dataset.

$$\begin{aligned} d(x) &= 26\times \mid shift_{-1,-1}(x) - shift_{-1,0}(x) \mid + \\ &\quad (\mid shift_{-1,0}(x) - \sqrt{x} \mid + \\ &\quad \mid shift_{0,-1}(x) - \sqrt{x} \mid)^2 \end{aligned} \tag{4}$$

Here $d(x)$ is the output of the detector and the pixel is classified as an edge if $d(x)$ is positive. Since we are dealing with binary images and only concerned with the sign of $d(x)$, this expression can be simplified to (5).

$$\begin{aligned} d(x) &= \mid shift_{-1,-1}(x) - shift_{-1,0}(x) \mid + \\ &\quad \mid shift_{-1,0}(x) - x \mid + \\ &\quad \mid shift_{0,-1}(x) - x \mid \end{aligned} \tag{5}$$

We see that $\mid shift_{-1,0}(x) - x \mid$ and $\mid shift_{0,-1}(x) - x \mid$ are intensity differences to the left and up separately from the current pixel, and $\mid shift_{-1,-1}(x) - shift_{-1,0}(x) \mid$ allows for detection of the corner points after combining the other two parts. Therefore the detector can work well for the special binary image dataset featuring simple shapes. Taking some patterns from Fig 10 as an example, (a) and (c) are not edge points and (b) and (d) are edge points, and the detector in (5) will correctly mark these pixels. This detector only uses pixels

TABLE VI
COMPARISON BETWEEN F_m AND F_n FOR TEN TEST IMAGES FROM BSD

Detector		143090	175032	24077	253027	253055	299086	361010	42049	296059	3096
F_m	F	0.7210	0.9417	0.8155	0.8480	0.6582	0.7777	0.7331	0.4922	0.6182	0.5228
	s	0.9606	0.6275	0.8707	0.8534	0.9886	0.9733	0.9198	0.9671	0.9643	0.9910
F_n	F	0.7371	0.9424	0.8176	0.8583^-	0.6729	0.7951^-	0.7405	0.5104	0.6382	0.5379
	s	0.9582	0.6369	0.8778	0.8537	0.9869	0.9696	0.9165	0.9690	0.9600	0.9900

within a 3×3 window, but more complex edge detectors are often evolved by GP but take much more space to present and explain.

VI. CONCLUSIONS

The goal of this paper was to develop a GP-based method for edge detection using the whole image as input, together with different fitness functions. This goal was successfully achieved by using the operator $shift_{n,m}$ and the F-measure and a new measure as fitness functions. Based on three different image datasets, novel edge detectors evolved by GP can compete with the Sobel detector in some special images. The proposed fitness function can make GP evolve detectors similar with those using the F-measure as fitness function. The analysis of the evolved program detectors reveals that GP can automatically evolve new models for edge detection which are very different from traditional window-based edge detectors, without preprocessing or postprocessing.

For future work, we will first increase the number of training images used and determine their impact upon effectiveness of the evolved detectors. We will also introduce multiple objects into the images and address the issue arising when a detector predicts an edge very close to (perhaps just one pixel away from) a true edge.

REFERENCES

[1] H. Trichili, M.-S. Bouhlel, N. Derbel, and L. Kamoun, "A survey and evaluation of edge detection operators: application to medical images," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, 2002.

[2] D. Marr and E. Hildreth, "Theory of edge detection," in *Proceedings of the Royal Society of London, Series B, Biological Sciences*, vol. 207, no. 1167, 1980, pp. 187–217.

[3] 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 Conference on Genetic and Evolutionary Computation (GECCO)*, 2005, pp. 795–802.

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

[5] K. Krawiec, D. Howard, and M. Zhang, "Overview of object detection and image analysis by means of genetic programming techniques," in *Frontiers in the Convergence of Bioscience and Information Technologies*, 2007, pp. 779–784.

[6] Y. Li, J. Ma, and Q. Zhao, "Two improvements in genetic programming for image classification," in *IEEE Congress on Evolutionary Computation (CEC)*, 2008, pp. 2492–2497.

[7] T. Kowaliw, W. Banzhaf, N. Kharma, and S. Harding, "Evolving novel image features using genetic programming-based image transforms," in *IEEE Congress on Evolutionary Computation (CEC)*, May 2009, pp. 2502–2507.

[8] T. Liddle, M. Johnston, and M. Zhang, "Multi-objective genetic programming for object detection," in *IEEE Congress on Evolutionary Computation (CEC)*, 2010, pp. 1–8.

[9] R. Moreno, D. Puig, C. Julia, and M. Garcia, "A new methodology for evaluation of edge detectors," in *Image Processing (ICIP), 2009 16th IEEE International Conference on*, 2009, pp. 2157–2160.

[10] A. Pinho and B. Almeida, "A review on edge detection based on filtering and differentiation," *Revista Do Detua*, vol. 2, no. 1, pp. 113–126, 1997.

[11] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, pp. 679–698, 1986.

[12] 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, Aug. 2002.

[13] P. Terry and D. Vu, "Edge detection using neural networks," in *Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, vol. 1, 1993, pp. 391–395.

[14] M. Kunt, "Edge detection: a tutorial review," in *IEEE International Conference on ICASSP'82*, vol. 7, 1982, pp. 1172–1175.

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

[16] J. R. Koza, "Survey of genetic algorithms and genetic programming," in *Microelectronics Communications Technology Producing Quality Products Mobile and Portable Power Emerging Technologies (WESCON)*, 1995, p. 589.

[17] M.-J. Willis, H. Hiden, P. Marenbach, B. McKay, and G. Montague, "Genetic programming: an introduction and survey of applications," in *Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)*, 1997, pp. 314–319.

[18] P. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 2, pp. 121–144, 2010.

[19] J. Koza, "Human-competitive results produced by genetic programming," *Genetic Programming and Evolvable Machines*, vol. 11, pp. 251–284, 2010.

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

[21] M. Ebner, "On the edge detectors for robot vision using genetic programming," in *Workshop SOAVE 97, Selbstorganisation von Adaptivem Verhalten*, 1997, pp. 127–134.

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

[23] 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 (EuroGP)*, 2001, pp. 246–255.

[24] 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, March 2006.

[25] 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.

[26] N. L. Fernández-García, A. Carmona-Poyato, R. Medina-Carnicer, and F. J. Madrid-Cuevas, "Automatic generation of consensus ground truth for the comparison of edge detection techniques," *Image Vision Comput.*, vol. 26, pp. 496–511, 2008.

[27] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, 2001, pp. 416–423.