Genetic Programming for Extracting Edge Features Using Two Blocks

Wenlong Fu[1] **Mengjie Zhang**[2] **Mark Johnston**[3]

[1] School of Mathematics, Statistics and Operations Research
Victoria University of Wellington, PO Box 600, Wellington, New Zealand
Email: wenlong.fu@msor.vuw.ac.nz

[2] School of Engineering and Computer Science
Victoria University of Wellington, PO Box 600, Wellington, New Zealand
Email: mengjie.zhang@vuw.ac.nz

[3] Institute of Science and the Environment
University of Worcester, Henwick Grove, Worcester WR2 6AJ, United Kingdom
Email: m.johnston@worc.ac.uk

Abstract

In low-level edge detection, single pixels have been popularly used to extract edge features. However, the extracted edge features might not have good ability to effectively mark edge points on images with noise or/and textures. Single pixels can be extracted based on a local window. To automatically search pixels to extract edge features using Genetic Programming, search operators based on single pixels and single blocks of pixels have been proposed. Single blocks of pixels can be used to improve detection performance on natural images, but the computational cost is high. In this paper, to reduce the computational cost of using blocks of pixels, a new search operator based on two blocks of pixels is proposed. The experiment results show that the proposed search operator can effectively reduce computational cost on evolved edge detectors, remaining good detection performance. *Keywords:* Genetic Programming, Edge Detection, Feature Extraction

1 Introduction

Edge detection is a subjective task, and has been investigated more than three decades (Kunt 1982, Papari & Petkov 2011). In general, there are three stages in edge detection: pre-processing, feature extraction and post-processing. Feature extraction is an import stage. Techniques for pre-processing, such as filtering, and post-processing, such as thinning edges, can be routinely cooperated with feature extraction approaches (Martin et al. 2004, Moreno et al. 2009*a*).

In low-level edge detection, pixels from local windows are extracted to construct edge features for discriminating pixels as edge points or non-edge points. Window size is a trade-off between detection accuracy and localisation (Basu 2002). To avoid using windows, search operations need to be based on full individual images. Since edge features are implicit, there are no generic approaches to extracting edge features from images. The intensity value of a single pixel is often not sufficient to discriminate that pixel as an edge point or a non-edge point. To automatically search for neighbouring pixels for constructing edge features, Genetic Programming (GP) has been applied to edge detection (Harris & Buxton 1996, Poli 1996, Fu et al. 2011*b*).

However, it is difficult to suppress textures using single pixels (Papari & Petkov 2011, Ganesan & Bhattacharyya 1997). From the results in (Harris & Buxton 1996, Poli 1996, Fu et al. 2011*b*), detected results are also affected by noise and some textures. From human observation, boundary information requires a reasonable area surrounding the boundary to be recognised, so features extracted from small areas (based on blocks) are useful to detect edges via filtering noise and textures (Martin et al. 2004, Papari & Petkov 2011). To suppress noise and textures, single blocks of pixels have been proposed to extract edge features by GP (Fu et al. 2012*b*). From the results, it is effective to use single blocks of pixels to improve detection accuracy. However, it is found that the computational cost of the evolved edge detectors for detecting images is obviously increased, comparing to the evolved edge detector based on single pixels only. It is worth further investigating how to reduce computational cost and remain detection accuracy when GP utilises search operators based on blocks of pixels to extract edge features.

1.1 Goals

The goal of this paper is to reduce the computational cost of GP using blocks of pixels to evolve edge detec-

tors to extract edge features while remaining detection accuracy. Little prior domain edge knowledge, only using training images and their ground truth, is provided for automatic edge feature extraction. In the existing work (Fu et al. 2012*b*), single blocks of pixels were combined to construct GP edge detectors. There are a large number of candidate combinations on the proposed blocks. However, there are also many combinations which are not useful for extracting edge information. To reduce the number of candidates and remove some useless combinations, a new search operator based on two blocks (a simple combination of two single blocks) is proposed. Specifically, the following research objectives will be investigated.

- Whether constructing search operators using two blocks of pixels is better than using a single block of pixels to evolve low-level edge detectors, in terms of detection accuracy.
- Whether constructing search operators using two blocks of pixels is faster than using a single block of pixels to evolve low-level edge detectors.

Different from the existing work (Fu et al. 2012*b*), we further investigate the searching (pixels) behaviour from GP. We utilise different combinations of search operators to evolve edge detectors.

1.2 Organisation

In the remainder of this paper, Section 2 briefly describes edge detection background and presents existing work in edge detection using GP with discussions. Section 3 introduces the GP system using blocks of pixels. Section 4 describes the experiment design. After the experiment results with discussions are presented in Section 5, conclusions are drawn and potential future investigation is suggested in Section 6.

2 Background

Relevant background on edge detection is briefly described in this section. Then, we give a brief survey on GP for edge detection.

2.1 Edge Detection

Edge features are used to discriminate pixels as edge points or not. In general, in the pre-processing stage, techniques are designed to filter noise and suppress textures; and in the post-processing stage, techniques are introduced to thin edges, remove standalone edge points, and link broken edges (Papari & Petkov 2011). The stages of pre-processing and post-processing are commonly employed to different edge detectors (Lopez-Molina et al. 2013). Therefore, edge features extracted by edge detectors usually determine the detectors' performance (Moreno et al. 2009*a*). In (Martin et al. 2004), different boundary detectors are trained by combining the same set of basic features, and those learnt detectors have similar results.

From the way of extracting edge features, we can extract edge features from low-level, mid-level and high-level. Generally, image context or object knowledge can be used to extract mid-level or/and high-level edge features. For instance, the Gestalt Laws (Papari & Petkov 2008) and the objects' similarity (Bai et al. 2008) has been used for edge feature extraction. Also, high-level features can be combined from low-level features. Low-level features are directly extracted from raw pixels in a local area, and they do not consider the image content.

In the early stage of the edge detection development, the edge feature extraction approaches mainly focused on low-level edge feature extraction (Ganesan & Bhattacharyya 1997, Kunt 1982). "Edge" is considered discontinuities on one-dimensional or two-dimensional signals (Basu 2002). Methods using differentiation techniques are popular to obtain edge responses on discontinuities. Edge detectors have utilised the gradients as edge responses, such as the Prewitt and Sobel edge detectors (Ganesan & Bhattacharyya 1997). To filter noise, methods via combining pre-processing (Gaussian filtering) and differentiation are employed to extract edge features, such as the Canny edge detector (Canny 1986). From retinal receptive fields using Gaussian functions, Gaussian-based edge detectors have been widely developed (Basu 2002). A common computational framework, using single raw pixels in a small local window to extract edge features, was suggested for edge detection (Ganesan & Bhattacharyya 1997).

Window size is a trade-off between localisation and noise rejection. A small window can be used to find details of edges, but the detected results are easily affected by noise and textures (Papari & Petkov 2011, Basu 2002, Bertero et al. 1988). From (Bertero et al. 1988), it is shown that the computation of the derivatives of a digital image is an ill-posed problem because there are no unique solutions for derivatives. When using a window to calculate edge responses, a big window has problems for edge localisation, a small window has problems of rejecting noise.

To reject noise and suppress textures, blocks of pixels are used to extract edge features. The difference of two blocks of pixels are used to indicate edge responses, such as dissimilarity indication from statistical test techniques (Lim & Jang 2002). In (Lim & Jang 2002), different two-sample tests are employed to extract edge features. Besides directly using the intensities of pixels, the intermediate results of blocks of pixels are employed to suppress noise and textures, such as the surround suppression technique (Grigorescu et al. 2004). In the surround suppression technique, gradients and the outputs of an edge detector using Difference of Gaussians (DoG (Marr & Hildreth 1980)) are combined together for edge detection.

2.2 Related Work for Edge Detection Using GP

This subsection briefly surveys the related work for edge detection using GP. From the view of employing knowledge in GP for edge detection, GP approaches for edge detection can be categorised as low-level extraction methods (little edge knowledge) and combination methods with some edge knowledge.

2.2.1 Low-level Extraction

When only little edge knowledge is used in designed tasks of GP for evolving edge detectors, it is expected that the outputs of the evolved edge detectors using pixels have similar outputs from human design (marked edges or existing designed edge detectors). In the training stage, generally, the desired outputs from human design are used. To extract edge features, search operators using single pixels (shifting functions) have been employed for evolving detectors based on full images (Poli 1996, Fu et al. 2011*b*, 2012*c*,*b*). Terminals including bits of the pixels in a 4×4 window were used to evolve edge detectors which were used to design digital circuits for edge detection (Golonek et al. 2006). Also, edge responses are generated based on existing knowledge, and then they are approximated by GP. In (Harris & Buxton 1996), one-dimensional step signals and the relevant edge responds were designed. GP was utilised to evolve formulae to fit these responses. These formulae were utilised to design one-dimensional edge detectors. Responses from existing edge detectors are also approximated by GP, such as the outputs of the Sobel and Canny edge detectors (Ebner 1997, Hollingworth et al. 1999, Harding & Banzhaf 2008).

The advantage of GP evolving edge detectors with little knowledge is that the training stage is independent of the background of edge detection. Different from human design, new programs can be found by GP. Based on desired outputs, GP is flexible to evolve different edge detectors from specific tasks. However, the search space for these techniques is too large, how to efficiently and effectively search pixels to construct edge detectors is a remaining issue.

2.2.2 Extraction using Some Edge Knowledge

In order to efficiently construct edge detectors, some simple edge knowledge has been used. From analysing edges and non-edges in a local area, Zhang and Rockett (Zhang & Rockett 2005) summarised edge patterns and non-edge patterns with 13×13 windows, and then edge detectors using pixels from a 13×13 window as terminals were evolved. Some image operators also have been used to construct edge detectors. Morphological operators erosion and dilation as the terminal set were employed to evolve morphological edge detectors (Quintana et al. 2006, Wang & Tan 2010). Gaussian operators and other image operators were used to evolve a high-level feature by GP, and this edge feature was combined with other edge features as a trained logistic regression classifier to detect object boundaries (Kadar et al. 2009). Additionally, three basic features were employed to combined composite features by GP (Fu et al. 2012*a*, 2013*b*,*a*).

The advantage of using edge knowledge to evolve new edge detectors is that the performance of these evolved edge detectors is not too low. However, these methods are dependent on the used knowledge. When only training images and their ground truth are provided, how to effectively and efficiently search pixels to extract edge features still needs to be investigated.

3 GP System Based on Full Images

This section describes the new GP system modified from the existing work (Fu et al. 2012*b*).

3.1 Sets of Terminals and Functions

This GP system uses a full image as a terminal. In general, constants are helpful for constructing GP programs in many applications (Koza et al. 1999, Poli et al. 2008). Besides the full image I, the terminal set contains random constants. The range of random constants rnd is from -10 to 10 based on initial experiments.

For the function set, the common arithmetic operators include the addition ($+$), subtraction ($-$), multiplication ($*$), division ($\div$), absolute (abs), square ($square$) and square root ($sqrt$). All functions work on each element of a matrix, such as each pixel of the input image I. The $+, -, *$, abs, $square$ have their usual meanings. The square root function $sqrt$ is protected, which produces a result of 0 for negative inputs. Division $\div$ is also protected, producing a result of 1 for a 0 divisor.

In existing work, there are two proposed search operators as functions: one search operator $s_{n,m}$ (Fu et al. 2011*a*) based on a single pixel and the other search operator $block_{t,l,w,d}$ (Fu et al. 2012*b*) based on a single block of pixels.

3.1.1 Function $s_{n,m}$

Function $s_{n,m}$ shifts its argument (a single two-dimensional matrix input) by n columns and m rows. If n is positive, a right shifting operation performs on the input, otherwise a left shifting operation performs on its argument. If m is positive, the two-dimensional input shifts down, otherwise shifts up. Its argument can come from image I or an intermediate result of a subtree, $sub(I)$, constructed by the GP system.

Note that if the two-dimensional input is rnd, rnd is considered as a two-dimensional matrix with its elements being equal to a single random constant. The shifting operation performs on the bits of rnd, and its value is multiplied by 2^{n+m} so that the GP system can generate a large range of different constants. Here n and m are randomly selected from $\{-2, -1, 0, 1, 2\}$. When a new shifting function $s_{n,m}$ is generated and

11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55

(a) before calling $s_{-1,0}$

11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55

(b) after calling $s_{-1,0}$

Figure 1: Example two-dimensional matrix and its result after calling $s_{-1,0}$.

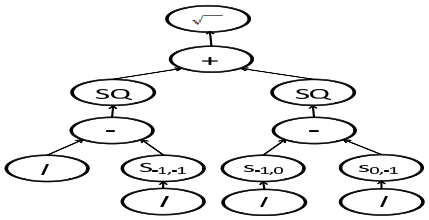


Figure 2: The 2×2 Roberts detector constructed with GP. Nodes "$\sqrt{\ }$" and "SQ" are functions *sqrt* and *square*, respectively.

its argument is image I, each pixel or one of its neighbours in a 5×5 window has equal probability to be selected.

Figure 1 (a) shows an example of a small two-dimensional matrix (for a ramp edge), and (b) is its result after calling a shifting function $s_{-1,0}$. Note that the last column of the shifted result is filled by the nearest element in the matrix. Via using shifting functions to implicitly search neighbours, neighbours of each discriminated pixel can be combined for constructing edge features with common operators.

It is possible for this GP system to generate some existing edge detectors. For instance, the 2×2 window Roberts detector (see Equation (3)) (Ganesan & Bhattacharyya 1997) is represented by the GP edge detector $GE_{Roberts}$, which is given by Equation (6), where $\circledast$ is a convolution operator. In order to employ the neighbours of each discriminated pixel (including each discriminated pixel itself) used in the Roberts detector, functions $s_{1,1}$, $s_{1,0}$ and $s_{0,1}$ are used to select the pixels around each discriminated pixel. Figure 2 presents a tree representation of the 2×2 window Roberts filter based on full image I.

$$R_{Roberts,x} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \circledast I \quad (1)$$

$$R_{Roberts,y} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \circledast I \quad (2)$$

$$R_{Roberts} = \sqrt{R^2_{Roberts,x} + R^2_{Roberts,y}} \quad (3)$$

$$GE_{Roberts,x} = I - s_{-1,-1}(I) \quad (4)$$

$$GE_{Roberts,y} = s_{-1,0}(I) - s_{0,-1}(I) \quad (5)$$

$$GE_{Roberts} = \sqrt{GE^2_{Roberts,x} + GE^2_{Roberts,y}} \quad (6)$$

3.1.2 Function $block_{t,l,w,d}$

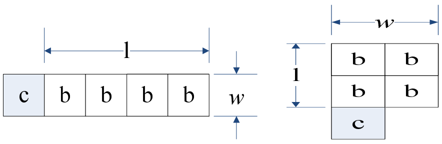
In general, it is a difficult task for GP to evolve a good edge detector using only single pixels to remove influence of noise and textures. The edge detector employs the intensity level of each single pixel, and it is sensitive to the pixel intensities. For instance, a pixel, which is not an edge point and is not affected by noise or texture, is easily marked as a non-edge point by an edge detector, such as the Sobel edge detector (Ganesan & Bhattacharyya 1997). After adding noise to the pixel, the edge detector might detect it as an edge point. In order to suppress noise and textures, information for an edge from a local area needs to be used. In general, a local area for indicating a pixel as an edge point includes a set of its neighbours.

If only $s_{n,m}$ is used to filter noise and textures, filters as subtrees need to be constructed, and then a program is constructed by these subtrees. However, the sizes of GP trees including filters are normally very large. If a large size tree is allowed in the GP system, the search space (tree size) will be exponentially increased, which leads to an increase in the computational cost. For example, the size of a full binary tree from depth $dp - 1$ to dp is increased by $O(2^{dp-1})$. Additionally, some existing filters cannot be constructed by the simple GP system, such as a median filter (Bovik et al. 1987).

To suppress textures and reject noise, existing work employs a set of pixels. The dissimilarity of two blocks of pixels was indicated by statistical approaches (Lim & Jang 2002). A surround suppression technique utilised blocks of pixels' intermediate results (gradients) to remove some texture responses (Grigorescu et al. 2004). To simulate this idea, new approaches to using a set of pixels or their intermediate results need to be developed so that the GP system has some ability to search blocks of pixels to reject noise and suppress textures.

Search operator $block_{t,l,w,d}$ is designed to find blocks of pixels to extract edge features so that the GP system can effectively construct edge detectors with some ability to filter noise and textures. The search operator $block_{t,l,w,d}$ includes approaches (t) to transforming a block of pixels to a single variable (such as the mean of intensities of all pixels in the block), the block size parameters l (the length of the block) and w (the width of the block), and the directional position d (where the block is located around a discriminated pixel). The argument of the search operator is a two-dimensional matrix, which is image I or an intermediate result from a subtree $sub(I)$. Figures 3 (a) and (b) describe two examples for the blocks of pixels specified by the search operator. Here, a block of pixels ("bbbb") is specified relative to the discriminated pixel ("c"). In Figure 3 (a), the parameters for the block are $l = 4$, $w = 1$ and $d =$"right"; and in Figure 3 (b), the parameters for the block are $l = 2$, $w = 2$ and $d =$"up".

The parameter t is used to transform a set of pixels to a single variable. The purpose of using a block of



(a) Right (b) Up

Figure 3: Two example blocks of pixels specified by search operator $block_{t,l,w,d}$.

pixels is to indicate some special characteristics from the set. Mean and standard deviation are commonly used to summarise a group of data. Therefore, mean ($t = 0$) and standard deviation ($t = 1$) are employed in the search operator $block_{t,l,w,d}$.

The search operator $block_{t,l,w,d}$ is used to find a set of pixels around a discriminated pixel, and it has no ability to move to a new position. From the comparison between $s_{n,m}$ and $block_{t,l,w,d}$, the latter only searches a limited area for constructing edge detectors. Without $s_{n,m}$, the pixels only found by the search operator $block_{t,l,w,d}$ are dependent on its parameters l, w and d. However, $s_{n,m}$ can search pixels which are far away from the relevant discriminated pixel, and the ability to find pixels is dependent on not only its parameters n and m, but also the maximum depth of a GP tree.

In contrast to the existing approaches to utilising blocks of pixels, the search operator $block_{t,l,w,d}$ provides four distinguished characteristics. Firstly, an unfixed size block (used to find a set of pixels) is different from the common way using a fixed block (Dollar et al. 2006, Martin et al. 2004, Papari & Petkov 2011). It is possible that GP can find a suitable size block to construct edge detectors, which decreases the computational cost but does not affect the ability to detect edges. Secondly, flexible positions to find a block of pixels are used after calling the search operator $s_{n,m}$. In a ramp edge or a stair edge, the closest neighbours might affect the detection results because of its discontinuity. Using a combination of $s_{n,m}$ and $block_{t,l,w,d}$, such as $block_{t,l,w,d}(s_{n,m}(I))$, possibly avoids this influence. Thirdly, different directional positions are used together to construct edge detectors so that the directional calculation is avoided. Lastly, a flexible input in $block_{t,l,w,d}$ comes from original image I or an intermediate result after some processing on image I (subtree). After calling a subtree $sub(I)$ for filtering noise or textures, $block_{t,l,w,d}(sub(I))$ might help improve detection performance.

Figure 4 shows an example of searching blocks of pixels for discriminating the centre pixel (blue) in four different directions. Here, the neighbours with the same colour are in the same block. There are two major areas with intensities 11 and 55 respectively. Three columns with intensities 20, 30 and 40 are the critical region between both major areas. From existing methods, a comparison between the right block and the left block or between the right block and the

11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55

Figure 4: Example of searching neighbours by $block_{t,l,w,d}$.

bottom block is easy to find the ramp edge. For instance, the mean of pixel intensities from different blocks can be compared. The difference of the means of pixel intensities in the left block and the right block is $\frac{11*4+20*2}{6} - \frac{40+55*3}{4} = -37.25$. Here, out of the right area is considered as the area with intensity 55. To discriminate the right neighbour (at the sixth column), all blocks move right by a pixel, and the right block has four "55" (three in the figure and one being out of the right area). The differences of the means for the right neighbours (the sixth, seventh and eighth columns) are -34.67 ($\frac{11*2+20*2+30*2}{6} - \frac{55*4}{4}$), -25.00, and -13.33, respectively. The differences of the means between the left block and the up block for the pixel and its relative right neighbours are -21.00, -27.17, -25.00 and -13.33, respectively. Since the means from the left block and the up block are close, the relative response on the ramp edge is thick.

In Figure 4, if only using $s_{n,m}$, a high threshold 14 might be used to detect the boundary between the seventh and sixth columns based on the difference of two columns. The difference of the intensities between the seventh and sixth columns is 15 ($55 - 40$). However, if the intensities of the pixels at the last three columns are changed to 50 and the intensities of the pixels at the first three columns are changed to 10, the responses from the third columns to seventh columns are the same ($30 - 20 = 40 - 30 = 50 - 40 = 10$).

3.2 New Search Function

While $block_{t,l,w,d}$ can be used to search good edge features as discussed previously, the search space is huge. To reduce the search space of using blocks of pixels, two blocks with the same size are combined by common operators, and a new search operator is developed for combining two blocks of pixels. In edge detection, edge points are often located at boundaries between "different" areas. The detection mainly focuses on finding differences. Subtraction $-$ and division $\div$ are usually used for indicating differences from numerics. Therefore, we employ $-$ and $\div$ in the new search operator tb_{j,t,l,w,d_1,d_2} ($j =$'$-$' or $j =$'$\div$') using two blocks of pixels. Equations (7) and (8) give the definition of the search operator tb_{j,t,l,w,d_1,d_2} for $j =$'$-$' and $j =$'$\div$', respectively. Here, ε is a small positive constant, d_1 and d_2 are different directions,

11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55

(a) horizontal direction

11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55

(b) vertical direction

Figure 5: Examples of using two blocks of pixels.

and $tb_{\div,t,l,w,d_1,d_2}$ requires that $block_{t,l,w,d_2}$ must be larger than or equal to 0.

$$tb_{-,t,l,w,d_1,d_2} = block_{t,l,w,d_1} - block_{t,l,w,d_2} \quad (7)$$

$$tb_{\div,t,l,w,d_1,d_2} = \frac{block_{t,l,w,d_1}}{block_{t,l,w,d_2} + \varepsilon} \quad (8)$$

Figures 5 (a) and (b) show combinations of two blocks of pixels used for tb_{j,t,l,w,d_1,d_2}. To discriminate the centre pixel, four combinations of two blocks are shown in Figure 5. Here, the same colour blocks are used in the same search operator tb_{j,t,l,w,d_1,d_2}. In this study, the combinations of d_1 and d_2 are limited to the four different combinations: left up and right up, left bottom and right bottom, left up and left bottom, and right up and right bottom.

Using tb_{j,t,l,w,d_1,d_2} is approximately considered as a small subset of using $block_{t,l,w,d}$. The aim of proposing tb_{j,t,l,w,d_1,d_2} is efficiently reducing the space for searching blocks of pixels while remaining detection accuracy. Note that tb_{j,t,l,w,d_1,d_2} is not randomly selected from combinations of $block_{t,l,w,d}$. There are two elements of prior knowledge used in tb_{j,t,l,w,d_1,d_2}, namely the comparison using difference and the approach (t) frequently used in the evolved edge detectors when $block_{t,l,w,d}$ is used. Additionally, when a random constant rnd is used as an argument in $block_{t,l,w,d}$ or tb_{j,t,l,w,d_1,d_2}, the return value is the same as rnd.

3.3 Fitness Function

The class label for edge detection only has "edge point" or "non-edge point" here, and the main class is "edge point". For the output of a program, 0 is employed as the threshold for discriminating a pixel as an edge point (larger than 0) or a non-edge point (less than or equal to 0), and all images use the pixel intensities. Since only low-level edge detectors are evolved in this study and searching operators are employed to find features to construct GP detectors (namely this study only focuses on the way to extract feature for edge detection), the output is directly evaluated without post-processing, following (Moreno et al. 2009*b*).

The F-measure technique (Martin et al. 2004) has been widely used for evaluating edge detectors' performance. The F-measure technique is a combination of recall and precision with a parameter factor α (from 0 to 1). Here, recall is the number of correctly predicted edge points as a proportion of the total number of pixels on the true edges, and precision is the number of correctly predicted edge points as a proportion of the total number of predicted edge points. The fitness function using F-measure technique is given in Equation (9). To balance recall and precision, α is set to 0.5 in this paper.

(a) image 23080 (b) GT of image 23080

(c) sampled result (d) GT of the sampling

Figure 6: Example training image 23080, its ground truth (GT) and its sampled result of size 125 × 125 pixels.

$$F = \frac{\text{recall} * \text{precision}}{\alpha * \text{recall} + (1 - \alpha) * \text{precision}} \quad (9)$$

4 Experiment Design

This section describes an image dataset and the settings for the GP system.

4.1 Image Dataset

To evolve subjective edge detectors, the Berkeley Segmentation Dataset (BSD) (Martin et al. 2004) including natural images with ground truth provided are chosen. To sample training images as the training data, a small training dataset is employed. This image data set contains 20 BSD training images. The 20 images are images 42078, 106020, 68077, 23080, 216053, 61060, 41004, 113044, 134008, 161062, 163014, 189011, 207056, 236017, 249061, 253036, 271031, 299091,311081, and 385028. For each image, a subimage of size 125 × 125 pixels are randomly extracted. Figure 6 shows training image 23080 and its randomly sampled subimage of size 125 × 125 pixels. The test dataset is the 100 BSD full sized (481 × 321) test images.

4.2 Sets of Terminals and Functions

To investigate the influence of the function set, three settings (search operators) are used different combinations of the three search operators for the GP system. The first setting is from the existing work (Fu

et al. 2012*b*), namely including search functions $s_{n,m}$ and $block_{t,l,w,d}$. We use $Set_{s,b}$ to indicate the GP system including $s_{n,m}$ and $block_{t,l,w,d}$. The second setting uses the proposed search function tb_{j,t,l,w,d_1,d_2} and function $s_{n,m}$. we use $Set_{s,tb}$ to indicate the GP system including $s_{n,m}$ and tb_{j,t,l,w,d_1,d_2}. Set_s is used to indicate the GP system only using $s_{n,m}$. Note that the other functions and terminals are used with these three settings.

4.3 Parameter Settings

The parameter values for $block_{t,l,w,d}$ are: $t = 0$ for the mean or $t = 1$ for the standard deviation, l is from 3 to 7, w is from 1 to 7, and d is one of left, right, up or down direction. From the initial experiment results, standard deviation has higher occurrences than mean in the evolved edge detectors, so only standard deviation is used in tb_{j,t,l,w,d_1,d_2}. The parameter values for GP are: population size 800; maximum depth (of a program) 10; maximum generation 200; and probabilities for mutation 0.15, crossover 0.80 and elitism (replication) 0.05. These values are chosen based on common settings and initial experiments. Each GP experiment for each setting is repeated for 30 independent runs.

5 Results and Discussions

In order to compare this GP system with the existing methods using a moving window, a common system (Zhang & Rockett 2005) uses all pixels in a 5×5 window as terminals, and employs all functions in the proposed GP system, except for the three search operators. Note that n and m in $s_{n,m}$ are from -2 to 2. Using a single $s_{n,m}$ is equal to selecting a pixel from a 5×5 window. Therefore, the 5×5 window is chosen in the common system. The common system detects edge points based on raw pixels, not full images. In the common system, fixed neighbours are given as terminals. Like the Sobel edge detector, the common system moves the window pixel by pixel to detect edge points. $Set_{5\times5}$ is used to indicate the evolved edge detectors using the 5×5 window to detect pixels one by one. Note that this paper mainly compares the proposed GP system with the GP system in (Fu et al. 2012*b*).

5.1 Overall Results

Table 1 gives the test performance of the means and standard deviations of F, the maximum F of the evolved edge detectors, and the means and standard deviations of the test time on the 100 BSD test images. Here the test time on each image is on a single machine with CPU 3.1 GHz based on an implementation in C++. From Table 1, the highest mean of F comes from the edge detectors evolved by GP using $Set_{s,b}$ (the search operators $s_{n,m}$ and $block_{t,l,w,d}$). The common method using the 5×5 moving window ($Set_{5\times5}$) has the lowest mean of F. From the comparison of the best evolved edge detector (maximum F) from each setting, $Set_{s,b}$ has the best edge detector with $F = 0.3170$ in Table 1, which is increased by 21% ($\frac{0.3170-0.2619}{0.2619}$), compared with the best edge detector from $Set_{5\times5}$. The F values of the best edge detectors from GP using search operators with a single block of pixels are higher than 0.3. However, the F values of the best evolved edge detectors using single pixels are lower than 0.3.

For the computational cost, the evolved edge detectors using single pixels (less than 0.05 second) are much faster than the evolved edge detectors including the search operators with blocks of pixels (their average being longer than 0.1 second). Compared with the evolved edge detectors from $Set_{s,b}$, the evolved edge detectors from $Set_{s,tb}$ (using the search operator tb_{j,t,l,w,d_1,d_2}) obviously reduce the computational cost of detecting images. However, the average time of these edge detectors from $Set_{s,tb}$ is longer than the average time of the edge detectors from Set_s. In $Set_{s,tb}$, 18 of the 30 evolved edge detectors take obviously less than 0.1 second to detect a BSD image. Therefore, the search operator using two blocks of pixels effectively reduces the computational cost, compared to those using a single block of pixels. Note that a GP edge detector executes from the bottom up, which leads to redundancies existing in the calculations, such as the same subtrees. This is a potential reason that the computational cost in the evolved edge detectors using blocks of pixels is heavy.

The standard deviations in these evolved edge detectors using blocks of pixels (on detecting time) are large. If using $block_{t,l,w,d}$, the detecting time for evolved edge detectors is varied. From the detecting times in Table 1, it is found that combining tb_{j,t,l,w,d_1,d_2} with $s_{n,m}$ can improve the stability of the test times when blocks of pixels are used.

Note that the standard deviations of F in Table 1 are small, although the test times are varied. Since using $block_{t,l,w,d}$ is not stable for the detecting cost, all evolved edge detectors using $block_{t,l,w,d}$ cannot be considered as heavy computational cost detectors. In these experiments, some evolved edge detectors using $block_{t,l,w,d}$ take less than 0.1 second to detect a BSD test image. For instance, one evolved edge detector from $Set_{s,b}$ with $F = 0.2894$ only takes 0.04 seconds for detecting a BSD image.

5.2 Statistical Comparisons

Table 2 gives p-values using two-sample t-tests for each pair of settings. Here, the first group comes from the relevant setting in the first column and the second group comes from the relevant setting in the first row; $\downarrow$ indicates that the first group (in the first column) is significantly worse than second group (in the first row); and $\uparrow$ indicates that the first group is significantly better than the second group when using the significance level 0.05. Note that each setting has 30 independent runs. From the table, the evolved edge

Table 1: Test performance (mean ± standard deviation) of the GP edge detectors on the 100 BSD test images. Note that the time is for testing one BSD image.

	F	maximum (F)	time (seconds)
$Set_{5\times5}$	0.2563 ± 0.0046	0.2619	0.0403 ± 0.0140
Set_s	0.2632 ± 0.0098	0.2807	0.0213 ± 0.0093
$Set_{s,b}$	0.2953 ± 0.0161	0.3170	0.2971 ± 0.1939
$Set_{s,tb}$	0.2895 ± 0.0103	0.3063	0.1031 ± 0.0566

Table 2: Statistical p-values (two-samples t-tests) among constructed GP edge detectors from $Set_{5\times5}$, Set_s, $Set_{s,b}$, $Set_{s,tb}$ on the 100 BSD test images.

	Set_s	$Set_{s,b}$	$Set_{s,tb}$
$Set_{5\times5}$	$0.0011 \downarrow$	$0.0000 \downarrow$	$0.0000 \downarrow$
Set_s		$0.0000 \downarrow$	$0.0000 \downarrow$
$Set_{s,b}$			0.1106

detectors from Set_s are significantly better than the evolved edge detectors from $Set_{5\times5}$. It seems that this GP system using the search operator $s_{n,m}$ is better than the common approach to using the 5×5 moving window. From the p-values of comparing Set_s with the other settings (using blocks of pixels) respectively, the evolved edge detectors using blocks of pixels are significantly better than the evolved edge detectors using single pixels only. There is non-significant difference between $Set_{s,tb}$ and $Set_{s,b}$. From Table 2, the evolved edge detectors using blocks of pixels are significantly better the evolved edge detectors only using single pixels (Set_s and $Set_{5\times5}$). Based on the comparison between $Set_{s,b}$ and $Set_{s,tb}$, replacing $block_{t,l,w,d}$ with tb_{j,t,l,w,d_1,d_2}, GP evolves the edge detectors which remains detection performance, in terms of F.

From Tables 1 and 2, it is suggested that the search operator tb_{j,t,l,w,d_1,d_2} can improve efficiency for constructing edge detectors, while keeping the accuracy at the same time.

5.3 Visual Results

Figure 7 shows an example image detected by the best edge detectors using single pixels and blocks of pixels ($Set_{s,b}$, $Set_{5\times5}$, $Set_{s,b}$ and $Set_{s,tb}$). From the edge detectors using single pixels only, namely from $Set_{s,b}$ and $Set_{5\times5}$, the detected results are affected by noise and textures in image 385039. When search operators based on blocks of pixels are used, the detected results from $Set_{s,b}$ and $Set_{s,tb}$ are not strongly affected by textures. Comparing the edge detector from $Set_{s,tb}$ to the edge detector from $Set_{s,b}$, we find that there are no obvious difference. It seems that all of the GP evolved edge detectors using blocks of pixels suppress textures and filter noise. Compared with the edge detectors using single pixels only, the edge detectors using blocks of pixels do not decrease obviously in finding true edge points, but obviously remove lots of falsely predicted edge points, in terms of the detected visual results. The edge evolved by GP with $Set_{s,tb}$ are thicker than those with $Set_{s,b}$, but this can be easily thinned by a simple post-processing technique.

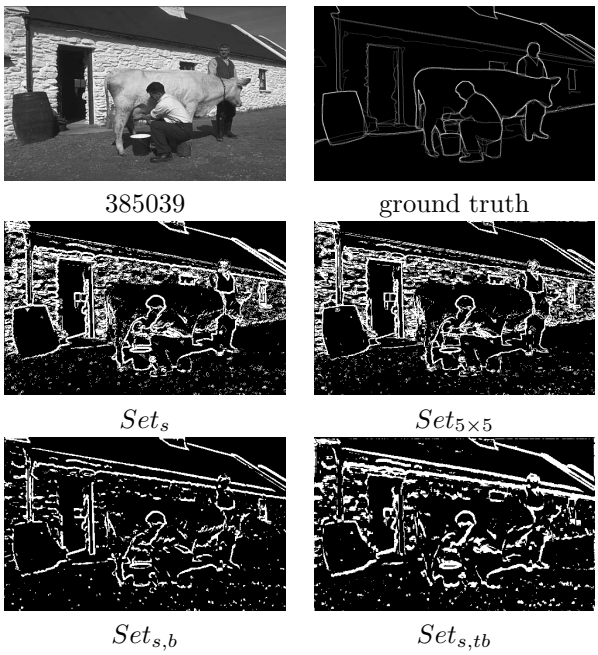


Figure 7: Example test images detected by the best edge detectors from Set_s $Set_{5\times5}$, $Set_{s,b}$ and $Set_{s,tb}$, respectively.

Note that when an offset is not allowed between a predicted edge point to a true edge point, recall and precision usually are not large. The difference of F between Set_s and $Set_{s,b}$ is small, but the visual detected results from them are obviously different.

6 Conclusions

The goal of this paper was to reduce the computational cost of GP using blocks of pixels to extract edge features while remaining detection accuracy. A search operator based on two blocks of pixels was proposed to reduce the number of potential combinations of the existing search operator based on a single block of pixels. From the experiment results, the GP system using the proposed new search operator based on two blocks of pixels can be used to reduce the computational cost on the evolved edge detectors while

remaining detection accuracy. It seems that reducing the search space on blocks of pixels is efficient to improve the performance of the evolved detectors, in terms of the computational cost.

For future work, the investigation on building blocks for edge detection will be conducted so that we can understand how GP works on construction of edge detectors. The computational complexity of using varying blocks in GP will be investigated. Also, different performance evaluation criteria will be employed to compare GP evolved edge detectors with existing edge detectors.

References

Bai, X., Yang, X. & Latecki, L. J. (2008), 'Detection and recognition of contour parts based on shape similarity', *Pattern Recognition* **41**(7), 2189–2199.

Basu, M. (2002), 'Gaussian-based edge-detection methods: a survey', *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* **32**(3), 252–260.

Bertero, M., Poggio, T. & Torre, V. (1988), 'Ill-posed problems in early vision', *Proceedings of the IEEE* **76**(8), 869–889.

Bovik, A., Huang, T. S. & Munson, D.C., J. (1987), 'The effect of median filtering on edge estimation and detection', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-9**(2), 181–194.

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

Dollar, P., Tu, Z. & Belongie, S. (2006), Supervised learning of edges and object boundaries, *in* 'Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition', Vol. 2, pp. 1964–1971.

Ebner, M. (1997), On the edge detectors for robot vision using genetic programming, *in* 'Proceedings of Horst-Michael Groβ, Workshop SOAVE 97 - Selbstorganisation von Adaptivem Verhalten', pp. 127–134.

Fu, W., Johnston, M. & Zhang, M. (2011*a*), Genetic programming for edge detection: a global approach, *in* 'Proceedings of the 2011 IEEE Congress on Evolutionary Computation', pp. 254–261.

Fu, W., Johnston, M. & Zhang, M. (2011*b*), Genetic programming for edge detection based on accuracy of each training image, *in* 'Proceedings of the 24th Australasian Joint Conference on Artificial Intelligence', pp. 301–310.

Fu, W., Johnston, M. & Zhang, M. (2012*a*), Automatic construction of invariant features using genetic programming for edge detection, *in* 'Proceedings of the Australasian Joint Conference on Artificial Intelligence', pp. 144–155.

Fu, W., Johnston, M. & Zhang, M. (2012*b*), Genetic programming for edge detection using blocks to extract features, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference', pp. 855–862.

Fu, W., Johnston, M. & Zhang, M. (2012*c*), Genetic programming for edge detection via balancing individual training images, *in* 'Proceedings of the IEEE Congress on Evolutionary Computation', pp. 2597–2604.

Fu, W., Johnston, M. & Zhang, M. (2013*a*), Genetic programming for edge detection using multivariate density, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference', pp. 917–924.

Fu, W., Johnston, M. & Zhang, M. (2013*b*), Triangular-distribution-based feature construction using genetic programming for edge detection, *in* 'Proceedings of the IEEE Congress on Evolutionary Computation', pp. 1732–1739.

Ganesan, L. & Bhattacharyya, P. (1997), 'Edge detection in untextured and textured images: a common computational framework', *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* **27**(5), 823–834.

Golonek, T., Grzechca, D. & Rutkowski, J. (2006), Application of genetic programming to edge detector design, *in* 'Proceedings of the International Symposium on Circuits and Systems', pp. 4683–4686.

Grigorescu, C., Petkov, N. & Westenberg, M. A. (2004), 'Contour and boundary detection improved by surround suppression of texture edges', *Image and Vision Computing* **22**(8), 609–622.

Harding, S. & Banzhaf, W. (2008), 'Genetic programming on GPUs for image processing', *International Journal of High Performance Systems Architecture* **1**(4), 231–240.

Harris, C. & Buxton, B. (1996), Evolving edge detectors with genetic programming, *in* 'Proceedings of the First Annual Conference on Genetic Programming', pp. 309–314.

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

Kadar, I., Ben-Shahar, O. & Sipper, M. (2009), Evolution of a local boundary detector for natural images via genetic programming and texture cues, *in* 'Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation', pp. 1887–1888.

Koza, J., Bennett III, F. H., Andre, D. & Keane, M. A. (1999), *Genetic Programming III: Darwinian Invention and Problem Solving*, Morgan Kaufmann.

Kunt, M. (1982), Edge detection: a tutorial review, *in* 'Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing', Vol. 7, pp. 1172–1175.

Lim, D. H. & Jang, S. J. (2002), 'Comparison of two-sample tests for edge detection in noisy images', *Journal of the Royal Statistical Society. Series D (The Statistician)* **51**(1), 21–30.

Lopez-Molina, C., De Baets, B. & Bustince, H. (2013), 'Quantitative error measures for edge detection', *Pattern Recognition* **46**(4), 1125–1139.

Marr, D. & Hildreth, E. (1980), 'Theory of edge detection', *Proceedings of the Royal Society of London, Series B, Biological Sciences* **207**(1167), 187–217.

Martin, D., Fowlkes, C. & Malik, J. (2004), 'Learning to detect natural image boundaries using local brightness, color, and texture cues', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**(5), 530–549.

Moreno, R., Puig, D., Julia, C. & Garcia, M. (2009*a*), A new methodology for evaluation of edge detectors, *in* 'Proceedings of the 16th IEEE International Conference on Image Processing (ICIP)', pp. 2157–2160.

Moreno, R., Puig, D., Julia, C. & Garcia, M. (2009*b*), A new methodology for evaluation of edge detectors, *in* 'Proceedings of the 16th IEEE International Conference on Image Processing', pp. 2157–2160.

Papari, G. & Petkov, N. (2008), 'Adaptive pseudo dilation for gestalt edge grouping and contour detection', *IEEE Transactions on Image Processing* **17**(10), 1950–1962.

Papari, G. & Petkov, N. (2011), 'Edge and line oriented contour detection: state of the art', *Image and Vision Computing* **29**, 79–103.

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

Poli, R., Langdon, W. B. & McPhee, N. F. (2008), *A Field Guide to Genetic Programming*, Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`. With contributions by J. R. Koza.

Quintana, M. I., Poli, R. & Claridge, E. (2006), 'Morphological algorithm design for binary images using genetic programming', *Genetic Programming and Evolvable Machines* **7**, 81–102.

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

Zhang, Y. & Rockett, P. I. (2005), Evolving optimal feature extraction using multi-objective genetic programming: a methodology and preliminary study on edge detection, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference', pp. 795–802.