Soft Comput (2016) 20:1231–1248
DOI 10.1007/s00500-014-1585-1

METHODOLOGIES AND APPLICATION

Genetic programming for edge detection: a Gaussian-based approach

Wenlong Fu · Mark Johnston · Mengjie Zhang

Published online: 10 January 2015

Abstract Gaussian-based filtering techniques have been popularly applied to edge detection. However, how to effectively tune parameters of Gaussian filters and how to effectively combine different Gaussian filters are still open issues. In this study, a new genetic programming (GP) approach is proposed to automatically tune parameters of Gaussian filters and automatically combine different types of Gaussian filters to extract edge features. In general, it is time-consuming for GP to evolve edge detectors using a large training image dataset. To efficiently evolve edge detectors from a large training image dataset, we propose sampling techniques (randomly selecting training images) to evolve Gaussian-based edge detectors. A sampling technique only using part of a set of images obtains similar performance to the training data using all of these images. The evolved edge detectors from the proposed sampling technique perform better than the Gaussian gradient and rotation invariant surround suppression. Based on the analysis of GP evolving edge detectors, it is suggested that combining Gaussian filters should be based on different types of Gaussian filters, and the Gaussian gradient should be considered as a major filter in these combinations.

Keywords Edge detection · Genetic programming · Sampling · Feature extraction

Communicated by V. Loia.

W. Fu (✉) · M. Johnston
School of Mathematics, Statistics and Operations Research, Victoria University of Wellington, PO Box 600, Wellington, New Zealand
e-mail: wenlong.fu@msor.vuw.ac.nz

M. Zhang
School of Engineering and Computer Science, Victoria University of Wellington, PO Box 600, Wellington, New Zealand

1 Introduction

Edge detection is a well developed area of image analysis (Kunt 1982; Papari and Petkov 2011). Edge detection generally aims at finding discontinuities between different regions or between objects and background. In order to detect discontinuous changes in an image, many methods have been proposed (Basu 2002; Ganesan and Bhattacharyya 1997; Papari and Petkov 2011). However, since edge detection is a subjective task, there are usually several different solutions for one natural image based on human observation. It is appealing to automatically construct edge detectors for special tasks.

Gaussian filters are useful for image processing (Basu 2002; Papari and Petkov 2011). Gaussian-based edge detection techniques have been developed for many years, and some advantages regarding filtering noise exist in these techniques (Basu 2002). Different Gaussian-based approaches have been investigated based on a single Gaussian filter, such as Laplacian of Gaussian (LoG) (Marr and Hildreth 1980) and Canny edge detectors (Canny 1986), or multiple Gaussian filters, such as multi-scale edge detection (Papari and Petkov 2011) and surround suppression (SS) (Grigorescu et al. 2004). Techniques based on multiple Gaussian filters can improve detection accuracy (Papari and Petkov 2011).

However, there are problems in Gaussian-based approaches. First, it is difficult to manually set the parameters (scales) of Gaussian filters. Gaussian filters at different scales are often combined to detect edges. In multi-scale edge detection (Basu 2002; Song and Li 1998), the scales of Gaussian filters and the window size affect the detection performance, but the literature does not address how to effectively and automatically tune parameters of Gaussian filters. Existing work has addressed automatic setting of filter scales, but these techniques are dependent on specific images (Basu 2002; Lindeberg 1996). The window size problem (blurring edges in a large window and noise influence in a small window) still exists in Gaussian-based edge detection. It is desirable to develop automatic techniques for setting parameters of Gaussian filters.

Second, it is not clear how to effectively combine Gaussian filters. When multi-scale Gaussian filters are combined in a fixed order, the detection performance is often dominated by the first processed Gaussian filter. SS combines different filters without any fixed order to improve detection accuracy (Grigorescu et al. 2003, 2004). In SS, an operation, called *inhibition*, is used to suppress texture responses. In general, the response from a Difference of Gaussians (DoG) (Basu 2002) is used in the inhibited term, and the response on the gradient of a Gaussian filter is the inhibited context. In SS, Gabor filters are usually used (Grigorescu et al. 2004). Since a two-dimensional Gabor filter is the product of a Gaussian kernel function and a sinusoidal function and Gaussian filters can replace Gabor filters in SS, SS can still be considered as a kind of Gaussian-based edge detection. The main benefit of SS is to filter noise caused by textures (Papari and Petkov 2011). From SS, effectively combining Gaussian filters can improve detection performance. However, it is difficult to reduce the influence on irregular textures by SS. Therefore, it is desirable to investigate new ways of effectively combining Gaussian filters.

Genetic programming (GP) has been employed for edge detection since at least 1996 (Harris and Buxton 1996; Poli 1996). The existing work for constructing edge detectors mainly focuses on low-level edge detectors via choosing raw pixels (Fu et al. 2011, 2012d; Zhang and Rockett 2005), or a combination of image operators (Kadar et al. 2009). These works show that GP can evolve good edge detectors (Fu et al. 2012c; Kadar et al. 2009; Zhang and Rockett 2005), but most of these approaches used a small training dataset. When all individual images in a large dataset are used as the training data, the computational cost in the training stage is high. Kadar et al. (2009) randomly selected images from a large image dataset as training data to evolve a feature using GP. Although only one feature was used to combine with other existing features to train a logistic regression classifier in their work, efficiently selecting a small set of images from a large dataset seems promising to train GP edge detectors alone (without combination with other existing edge detection approaches). In our previous work (Fu et al. 2013a), 20 images were employed to effectively evolve Gaussian-based edge detectors by GP. However, the computational cost is heavy when all 20 training images are used. Therefore, in order to reduce the computational cost in the training stage, this paper further investigate how to effectively select training images to evolve edge detectors. Based on the proposed GP system (Fu et al. 2013a), we also investigate the influence of using different types of Gaussian filters for GP evolving Gaussian-based edge detectors.

The goal of this paper is to investigate how to effectively use prior Gaussian-based knowledge in GP for edge detection. Some Gaussian-based filters are utilised by GP to construct Gaussian-based edge detectors for performance improvement. A function (Fu et al. 2013a) is proposed for combining different Gaussian filters. Since the Gaussian-based techniques are given as prior knowledge, this investigation mainly focuses on how to effectively and efficiently evolve Gaussian-based edge detectors.

GP is a time-consuming algorithm to evolve programs when a large image dataset is used for training. In order to reduce the computational cost from a full-image dataset in the training phase, an initial investigation on the relationship between the detection performance and training images is conducted. In machine learning, one-shot learning algorithms (Fink 2004; Li et al. 2006; Miller et al. 2000) have utilised prior domain knowledge (such as distributions on the images selected from their raw datasets) to reduce the number of training images. Different from the one-shot learning algorithms, the raw training data used in this paper is considered as being unseen and the structures of evolved edge detectors can be very different. Specifically, the following research objectives will be investigated: (1) whether a small set of images can be used to train good Gaussian-based programs; and (2) whether using different types of Gaussian filters is better than using a single type of Gaussian filter only to evolve Gaussian-based programs.

In the remainder of the paper, Sect. 2 briefly describes relevant background on edge detection and using GP for edge detection. Section 3 proposes a GP system to evolve Gaussian-based edge detectors. After presenting the experiment design in Sect. 4 and the related results in Sects. 5, 6 provides the further discussions. Finally, Sect. 7 draws conclusions and suggests future work directions.

2 Background

This section briefly describes relevant background on Gaussian-based edge detection and existing work on edge detection using GP.

2.1 Gaussian-based edge detection

From human visual systems, Gaussian filters play an important role to recognize edges (Basu 2002). Gaussian-based filters have been developed to extract edge features (Marr and Hildreth 1980; Schunck 1987). To filter noise and detect edges, Gaussian filters and differentiation have been used for edge detection, such as the Difference of Gaussians (DoG) (Marr and Hildreth 1980) and Gaussian gradients (Canny 1986).

Equation (1) describes a one-dimensional Gaussian filter $g_\sigma(x)$ with a scale parameter σ. Using scale parameters σ_1, σ_2, DoG is defined in Eq. (2). As a second derivative filter, DoG approximates LoG well (Song and Li 1998). Although DoG decreases overall image contrast, it suppresses noise usually having a high spatial frequency (Marr and Hildreth 1980).

$$g_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{1}$$

$$DoG_{\sigma_1,\sigma_2}(x) = g_{\sigma_1}(x) - g_{\sigma_2}(x) \tag{2}$$

Canny detectors (Canny 1986) are derived from an optimal filter based on the local maxima resulting from the convolution of a filter with the signal affected by white noise in one dimension, which is approximated by the derivative of a Gaussian function (Basu 2002). For the step edge (in one dimension), the derived optimal filter can be approximated by the first derivative of a Gaussian function with the scale parameter σ (see Eq. (3)) (Basu 2002). Canny edge detectors use adaptive thresholding with hysteresis to eliminate breaking of edge contours, but they are slightly sensitive to weak edges and susceptible to spurious and unstable boundaries with non-significant change in intensity (Papari and Petkov 2011; Basu 2002).

$$\frac{\partial g_\sigma(x)}{\partial x} \approx -\frac{x}{\sigma^2} \exp\left(-\frac{x^2}{\sigma^2}\right) \tag{3}$$

Using obvious response changes from Gaussian filters with various scales, edges could be detected. A specific filter can be used to smooth a relevant subregion of an image. Since an image may include different types of edges and noise, filters are combined to detect edges. In multi-scale techniques, there are three directions to employ Gaussian filters with different scales. The first, called edge focusing (Bergholm 1987), proceeds from a coarse solution to a fine solution. A large scale (high σ) Gaussian filter is utilised for finding edges; then locations of edges are determined by a next smaller scale. It is hard to set small and large scales of filters, although the aim of using multi-scale filters is to avoid blurred edges from large scale Gaussian filters. It is hard to choose a threshold to obtain binary edges from a filter, too. The quality of the detected edges is usually determined by the threshold used in a large scale filter. The second is from fine to coarse (Lacroix 1990). The problem of localisation error still exists when coarse solutions are used to detect edges. Again, it is not clear to choose scales in filters. The third is to use adaptive Gaussian filters to detect edges (Bennamoun et al. 1995). Assuming that noise is a Gaussian distribution with a known variance, this method smooths areas using a large scale to filter out noise. However, the noise variance in real images has to be estimated.

Multi-scale Gaussian filter techniques are based on specific orders of using different scale filters. Different from multi-scale Gaussian filters techniques, one popular approach (without considering the special orders) combines independent filters together, such as SS (Grigorescu et al. 2003, 2004). SS is used to effectively filter noise caused by some regular textures (Papari and Petkov 2011). In summary, multiple Gaussian filters are utilised to filter noise and improve detection accuracy because a single filter is not generally sufficient to filter noise and detect edges. However, considering automatic of feature extraction, it is required to further investigate how to automatically combine filters and automatically tune the parameters of filters.

2.2 Related work for edge detection using GP

In our previous work, we have done some work in edge detection using GP with different degrees of prior domain edge knowledge (Fu et al. 2011, 2012a, c, d, 2013a, b, c). Based on the fitness functions, the methods using GP for edge detection can be categorised as methods for minimising the mean square error (MSE) and methods for maximising the detection accuracy.

2.2.1 Minimising the MSE

When a fitness function is based on minimising the MSE, the aim is to evolve a detector whose outputs are close to the outputs from a desired detector or a model whose outputs are close to ideal outputs (responses). Similar to symbolic regression problems, GP is used to approximate the ideal outputs (not detected results) in edge detection. In order to get optimal filters for one-dimensional step signals, GP evolved programs with good responses in step signals (Harris and Buxton 1996), and then these programs were used to approximate filters. In (Harris and Buxton 1996), one-dimensional step edge signals were manually designed. The position of the signal was considered as a terminal, and normal arithmetic operators, such as addition, subtraction, multiplication and division, were used as the set of functions. The performance of the evolved edge detectors are dependent on the understanding of edges when edges are considered as one-dimensional signals. Since edges are very subjective, it is hard to represent most edges by one-dimensional signals. Another way to approximate detectors is to learn outputs from existing detectors. The outputs from one detector are considered as the targets, and then GP is used to find similar detectors. An approximation of the Sobel detector with the terminal set containing nine pixels in a shifting window was used to design a hardware detector (Hollingworth et al. 1999). A terminal set containing pixels from a 3 × 3 shifting window was used to approximate the Sobel detector with Cartesian GP (CGP) (Harding and Banzhaf 2008). Based on one whole image, four shifting functions introduced by Poli (Poli 1996) were used to approximate a Canny detector (Ebner 1997). However, since these evolved edge detectors only approximate a target detector, it is difficult for them to outperform the target detector.

An advantage of using the MSE is that the outputs of an evolved edge detector are similar to the desired responses for edge points and non-edge points, but the desired responses are hard to obtain. The desired outputs normally come from an existing edge detector, therefore, the performance of evolved edge detectors is dependent on the existing desired edge detector.

2.2.2 Maximising the detection accuracy

To obtain good detection accuracy such as the overall detection accuracy for the edge points and non-edge points, recall is often used for finding edge points and precision for predicting edge points. Here recall is the number of pixels on the edges correctly detected as a proportion of the total number of pixels on the edges, and precision is the number of pixels on the edges correctly detected as a proportion of the total number of pixels detected as edge points. GP has been used as a binary classification problem. In (Quintana et al. 2006; Wang and Tan 2010), morphological operators erosion and dilation were used as the terminal set and the evolved detectors classify pixels as edge points or non-edge points with the fitness functions based on a combination of recall and the specificity (the proportion of non-edge points which are correctly identified). Zhang and Rockett (2005) used 13×13 windows as patterns to extract features via multi-objective GP, with objectives of Bayesian error, classification error and the number of nodes.

In (Golonek et al. 2006), a digital circuit as an edge detector was evolved by GP with the fitness function based on minimising errors from wrong classifications. For detecting boundaries, a GP individual as candidate boundary cue was combined with a texture gradient cue into a single detector on a trained logistic regression classifier (Kadar et al. 2009) based on the F-measure. The terminal set in (Kadar et al. (2009) only included input images, and the function set included matrix addition, matrix subtraction, convolution with symmetric kernels, and etc. The symmetric kernels were predefined by the texture gradients from Martin et al. (2004). Only one evolved program was reported in Kadar et al. (2009). Since the task in Kadar et al. (2009) was to detect boundaries, rather than edges, some degrees of specific edge domain knowledge were employed. Note that edge features could be further processed for boundary detection using knowledge on boundaries, such as linking broken edges, removing short edges (not true edges) and thinning edges (Martin et al. 2004; Papari and Petkov 2011).

In (Bolis et al. 2001), the number of true edge points correctly found was used as one part of the fitness function to evolve an ant to find contours. In our previous work, the fitness function based on the combination of recall and precision was proposed to balance training images to evolve low-level edge detectors (Fu et al. 2012d). Also, we utilised detection accuracy to evolve low-level edge detectors based on full images with proposed search operators (Fu et al. 2011, 2012c, d) and composite features from a set of predefined basic features (Fu et al. 2013b, c). The evolved composite features are better than the predefined basic features, in terms of detection accuracy.

In both categories of methods, the size of training dataset is small. However, when the size of the training dataset becomes large, the computational cost is greatly increased. A real image dataset usually contains a lot of images, and each image might contain a large number of pixels. In (Fu et al. 2013b, c), the training data is sampled based on the predefined proportions of true edge points and true non-edge points in each training image. Since the Gaussian-based GP system is based on full images, rather than using a set of predefined basic features, how to efficiently reduce the computational cost in the GP system based on full images needs to be investigated.

3 The approach

The proposed Gaussian-based GP system is introduced in this section. In this GP system, the parameters of Gaussian filters and the operations between different Gaussian filters will be automatically evolved.

3.1 Terminals based on Gaussian models

To rapidly find a Gaussian-based edge detector, the terminal set in the proposed GP system includes three types of Gaussian filters: the Gaussian gradient, LoG, and DoG. The Gaussian gradient filter $dg_\sigma(u, v)$ is shown in Eq. (4), where σ is the scale parameter, and (u, v) is an offset (horizontal and vertical directions) from each discriminated pixel. The LoG filter $ddg_\sigma(u, v)$ is defined in Eq. (5). The DoG filter $dog_\sigma(u, v)$ is shown in Eq. (6).

$$
\begin{aligned}
g_\sigma(u, v) &= \frac{1}{2\pi\sigma^2}\exp\left(-\frac{u^2+v^2}{2\sigma^2}\right)\\
\frac{\partial g(u, v)}{\partial u} &= -\frac{u}{2\pi\sigma^4}\exp\left(-\frac{u^2+v^2}{2\sigma^2}\right)\\
\frac{\partial g(u, v)}{\partial v} &= -\frac{v}{2\pi\sigma^4}\exp\left(-\frac{u^2+v^2}{2\sigma^2}\right)\\
dg_\sigma(u, v) &= \sqrt{\left(\frac{\partial g(u, v)}{\partial u}\right)^2 + \left(\frac{\partial g(u, v)}{\partial v}\right)^2} \qquad (4)
\end{aligned}
$$

$$ddg_\sigma(u, v) = \frac{u^2 + v^2 - 2\sigma^2}{2\pi\sigma^6} \exp\left(-\frac{u^2+v^2}{2\sigma^2}\right) \tag{5}$$

$$dog_\sigma(u, v) = g_\sigma(u, v) - g_{2\sigma}(u, v) \tag{6}$$

The terminal set also includes random constants rnd (real numbers) in the range from -10 to 10 based on initial experiments. In this terminal set, the scales σ (real numbers) of all Gaussian filters are randomly generated in the continuous range from 1 to 5. Let the large scale in the DoG be *double* the small one, so the scale range of all Gaussian filters is from 1 to 10. Therefore, the coarsest scale (from 2 to 10) covers the range from 3 to 6 as suggested in Bergholm (1987) so that it is possible to find more Gaussian filters.

3.2 Function set

In the multi-scale edge detection technique, it is quite complex to use a function to describe the operation for detecting edges from a coarse solution to a fine solution or from a fine solution to a coarse solution. However, SS uses an operation to easily present a combination of different Gaussian filters to detect edges (Grigorescu et al. 2004). Therefore, the operation is used here to construct new Gaussian-based edge detectors.

Since Gaussian filters in the terminal set can be considered as edge detectors, a simple function set, namely $\{+, -, *, \div, \mathbb{C}\}$ is chosen. Here, $\div$ is protected division, producing a result of 1 for a 0 divisor; and $\mathbb{C}$ is a combination function from SS, which takes two arguments. Let $f_1(x, y)$ and $f_2(x, y)$ be image intensities or outputs from subtrees for a pixel with position (x, y), $f_1(x, y)\mathbb{C}f_2(x, y)$ represents the operation of the combination function. There are two steps in $f_1(x, y)\mathbb{C}f_2(x, y)$. In the first step, for each pixel, the first argument $f_1(x, y)$ provides the neighbours of the pixel with position (x, y) in a local 7×7 window. The second argument $f_2(x, y)$ provides the values of the relative neighbours (in the 7×7 window) transformed by Eqs. (7) and (8). Here, u and v (u' and v') are horizontal and vertical offsets, and $\sum_{u',v'}$ is the sum of positive$N(u', v', f(x, y))$ based on the 7×7 window. Note that norm$N(u, v, f(x, y))$ will be 0 if $\sum_{u',v'}$ is equal to 0. In the second step, convolution of the values (7×7 window) from $f_1(x, y)$ and $f_2(x, y)$ is performed to return a value for $f_1(x, y)\mathbb{C}f_2(x, y)$.

$$\text{positive}N(u, v, f(x, y)) = \max\{f(x + u, v + y), 0\} \tag{7}$$

$$\text{norm}N(u, v, f(x, y)) = \frac{\text{positive}N(u, v, f(x, y))}{\sum_{u',v'} \text{positive}N(u', v', f(x, y))} \tag{8}$$

An existing surround suppression technique (Grigorescu et al. 2004) can be expressed by the Gaussian-based GP system as GG$_{\text{SS}}$ in Eq. (9), where σ_1 and σ_2 are scaling parameters, dg_{σ_1} and dog_{σ_2} are the results after applying Gaussian filters dg_{σ_1} and dog_{σ_2} to an image, and A and B are constants. For a pixel, $dg_{\sigma_1}\mathbb{C}dog_{\sigma_2}$ returns the convolution of dg_{σ_1} with norm$N(u, v, dog_{\sigma_2})$ in the 7×7 window. If a pixel is located in a flat area (such as within which the responses dog_{σ_2} of its neighbours are 0), the operation $dg_{\sigma_1}\mathbb{C}dog_{\sigma_2}$ will return 0 and GG$_{\text{SS}}$ only depends on dg_{σ_1} and B. For a texture pixel with a non-zero response, its neighbours have similar responses or edge responses. In the 7×7 window, the number of edge points is typically small, the influence from the edge responses is not obvious. The return value of $dg_{\sigma_1}\mathbb{C}dog_{\sigma_2}$ is almost the same as the dg_{σ_1} value of the pixel. Therefore, GG$_{\text{SS}}$ will suppress the responses from textures. When a pixel is on a true edge and its neighbours have texture responses, most of the neighbours normally have weaker responses dg_{σ_1} than the pixel. Therefore, the return value of $dg_{\sigma_1}\mathbb{C}dog_{\sigma_2}$ is obviously lower than the response dg_{σ_1}. GG$_{\text{SS}}$ has slightly lower response contrast than dg_{σ_1}.

$$\text{GG}_{\text{SS}} = dg_{\sigma_1} - A \times dg_{\sigma_1}\mathbb{C}dog_{\sigma_2} - B \tag{9}$$

3.3 Fitness function

In our previous work (Fu et al. 2012d), the F-measure was used as the fitness function. However, the F-measure is a time-consuming evaluation system if a limited offset distance for detecting pixels to true edge points is allowed. That is because an optimal matching operation is required (Martin et al. 2004). We also investigated fitness functions based on the localisation accuracy, namely Figure of Merit (FOM) (Fu et al. 2012b). FOM as a fitness function can evolve low-level edge detectors, so we choose FOM based on each training image as the fitness function in this GP system. FOM is defined in Eq. (10), where M is the number of training images, $N_{i,T}$ is the number of true edge points in image i, $N_{i,P}$ is the number of predicted edge points in image i, $Set_{i,P}$ is the set of all predicted edge points for image i, α is a weighting factor for detection localisation, and $d(j)$ is the distance from a predicted edge point j to the nearest true edge point in a ground truth edge map. Considering the overlap of a 3×3 window, α is usually set to $\frac{1}{9}$.

$$\text{FOM} = \frac{1}{M} \sum_{i=1}^{M} \left(\frac{1}{\max\{N_{i,T}, N_{i,P}\}} \sum_{j \in Set_{i,P}} \frac{1}{1 + \alpha d^2(j)} \right) \tag{10}$$

3.4 Sampling techniques

To use the original images as the training set and reduce the computational cost during the training stage, two different sampling techniques based on original images as the training data are proposed here. The first technique is to randomly

Algorithm 1 Sampling Technique

```
1: Initialise the number of training images $M = M_{\min}$ and generation $g = 0$.
2: Randomly select a set of images (from 1 to $M$) as the evaluation data, evaluate all individuals, update population, and $g = g + 1$.
3: If Mod$(g, g_{\text{period}}) = 0$, then $M = M + 1$, otherwise go to step 5.
4: If $M = M_{\max} + 1$, then $M = M_{\text{restart}}$.
5: If the terminate criteria are satisfied, go to step 6, otherwise go to step 2.
6: Output the best program as the solution.
```

select one image as the evaluation data at each generation; and the second technique is to randomly select a set of images as the training data at each generation.

In the second technique, at the beginning, the number of the training images is a small number $M_{\min}$, and after every g_{period} generations, the number is increased by 1. When the number of images is equal to $M_{\max}$, the number of training images will return to a fixed value M_{restart}. M_{restart} is used to reset the number of training images. It is possible that $M_{\min}$ training images can distinguish the detection performance of initialised edge detectors. After some generations, these training images may not be good enough to distinguish edge detectors' performance, so more images are required for evaluating evolved edge detectors. When $M_{\max}$ training images are used, some evolved edge detectors might be obviously better than the others in the population. Again, M_{restart} training images could distinguish the detection performance of these edge detectors. The second sampling technique is described in Algorithm 1, where Mod(g, g_{period}) means that g modulo g_{period}. Note that the first random sampling technique can be described by Algorithm 1 after using the fixed $M = 1$ and removing steps 3 and 4.

In summary, the main framework of the proposed GP approach is shown in Fig. 1. Gaussian filters and random constants are used as terminals in the proposed GP system. At each generation, the training images are obtained by a sampling technique.

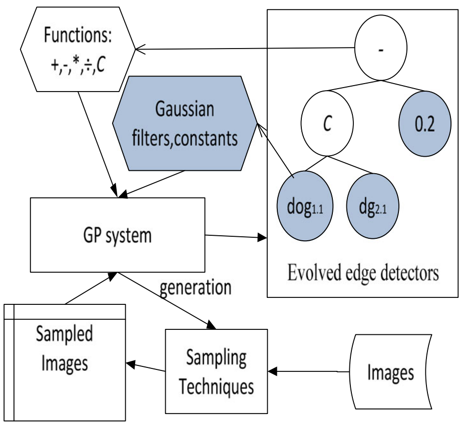

Fig. 1 The main framework of the proposed GP approach

4 Experiment design

We now describe the benchmark image dataset used in this paper. From the benchmark image dataset, we will select several (sub) sets of images as the training data.

4.1 Image datasets

The Berkeley Segmentation Dataset (BSD) (Martin et al. 2004) is popular for edge detection (Dollar et al. 2006), boundary detection (Kokkinos 2010) and image segmentation (Arbeláez et al. 2011). The BSD consists of natural images (of size 481 × 321 pixels) with ground truth provided. All images are independent and are taken from different places. The training dataset contains 200 images and the test dataset has 100 images. The ground truth on each image is combined from five to ten persons as a graylevel image for fairness of judgement of edges. Figure 2 shows six images from the training dataset and their ground truth. Note that the ground truth images are combined from several observations, therefore some edges are not one pixel wide only. The pixels with graylevel 0 (dark) in the ground truth are non-edge points, and the others are edge points.

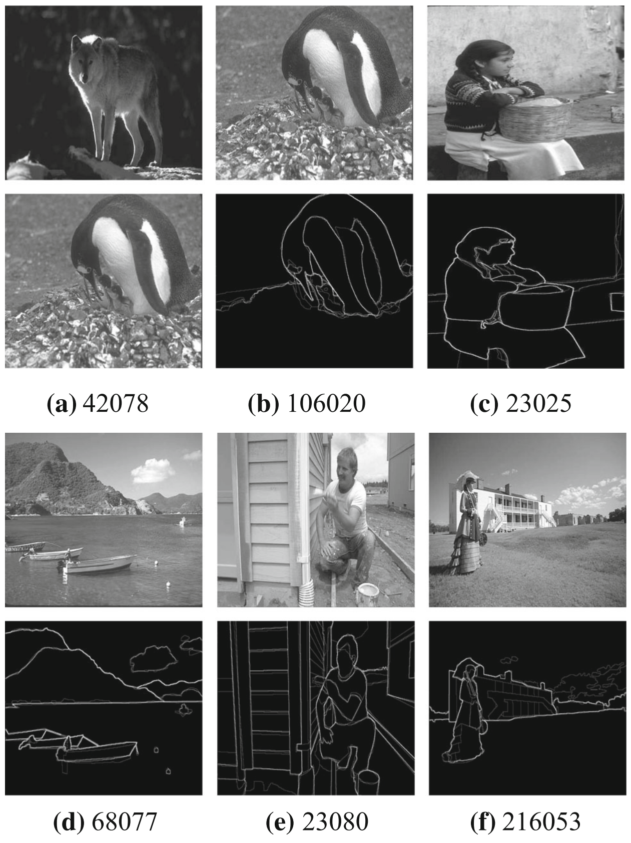

Fig. 2 Six example training images from the BSD dataset and their ground truth

In general, the performance evaluation of edge detection results is a challenging problem. The BSD dataset is based on segmentation results, and is not exact for true edges. However, edge detection is useful for boundary detection, and the results from edge detection affect the boundary detection results. Song et al. (2006) proposed to evaluate edge detection through boundary detection. In this paper, the evaluation for edge detectors evolved by GP is only related to the desired outputs, so we directly use the ground truth as the desired outputs to do the performance evaluation.

4.2 Experiment settings

Any Gaussian filter in the terminal set can be used to detect edges, therefore a new GP edge detector including Gaussian filters is expected to have some ability to perform edge detection. Since we use the fixed threshold 0 for all GP edge detectors, several images are employed to check whether a GP edge detector (with threshold 0) can detect true edge points. If all images are poorly detected, the GP edge detector usually performs poorly on other images. Therefore, we do not need to require a large number of images as training data. How many images are required and how to choose images as training data are investigated as follows, based on different images as training data.

4.2.1 Training data

To investigate the minimum number of training images required by GP, different sets of images are used as the training data. The first setting of the training data is to only employ a single full image. Each of the six images in Fig. 2 is selected as the training data respectively. These six images are selected based on different edge information, such as the single object in image 42078, irregular textures in image 106020 and many edges in image 23080. In terms of their evolved edge detectors' performance, the six images will be divided into three levels, namely the worst, middle and best levels. One-shot learning shows that a few of single training images can be used to learn new classifiers after combining prior knowledge from predefined features in training datasets (Li et al. 2006). Note that there are no fixed predefined features in the proposed GP system. Different from one-shot learning algorithms, this GP system does not use prior knowledge from training data. Without any prior edge domain knowledge, it is usually not sufficient to train edge detectors with a small set of training images. However, the Gaussian-based filters could be considered as prior edge knowledge here. A small set of training images is worth being investigated.

Second, in order to improve performance of the edge detectors evolved by a single training image, two images from the six images are selected to combine as the training data. It is possible that the training images in the best level are sufficient to train edge detectors, and the performance improvement mainly focuses on the edge detectors evolved from the worst and middle levels. The combinations of two images mainly come from the worst and middle levels. There are 15 combinations in total, but only four combinations are chosen. The four combinations are a combination of two images from the worst level, two combinations of one image from the worst level and one image from the middle level, a combination of one image from the middle level and one image from the best level.

We consider that 20 images are sufficient to train GP Gaussian-based edge detectors. In Fig. 2, all images, except for image 23025, are included in the 20 images. The other 15 images in the 20 images are images 61060, 41004, 113044, 134008, 161062, 163014, 189011, 207056, 236017, 249061, 253036, 271031, 299091,311081, 385028. S_{20} now is used for the whole 20 images. The reason to select image 23025 is that different edge information, such as the boundary between the girl and the background with different graylevels, exists in the image. Image 23025 is used to compare with S_{20} for training GP edge detectors.

Our previous work (Fu et al. 2012d) sampled 5 subimages (of size 51×51 pixels) from original images. Here, 5 subimages from each original image (of size 481×321 pixels) in S_{20} is also used as the training data. S_{sub} now will indicate the subimages sampling technique. Note that S_{rnd} now will indicate the first sampling technique based on the 20 full images S_{20}, S_{ada} now will indicate the second sampling technique based on S_{20}, and S_{20} represents the whole 20 images as the training data.

4.2.2 Parameter settings

The parameter values for GP are: population size 500; maximum generations 200; maximum depth (of a program) 7; and probabilities for mutation 0.15, crossover 0.80 and elitism (reproduction) 0.05. For the sampling technique S_{ada}, $M_{\text{restart}} = M_{\min} = 4$, $M_{\max} = 8$, and $g_{\text{period}} = 10$ are used. These values are chosen based on common settings and initial experiments (Espejo et al. 2010; Fu et al. 2011, 2012c, 2013a). There are 30 independent runs for each experiment.

The test performance evaluation is directly based on the binary outputs of GP edge detectors using the fixed threshold 0, without non-maximum suppression post-processing. All GP edge detectors are tested on the *same* 100 BSD test images. To measure the performance of GP edge detectors, the F-measure is used in the testing phase (Dollar et al. 2006; Martin et al. 2004). The F-measure (used in Dollar et al. 2006; Martin et al. 2004 as $F = \frac{2\text{recall} \times \text{precision}}{\text{recall}+\text{precision}}$) is a combination of recall and precision.

5 Results

The test results are separated into three parts: single images, two images, and a set of images. Since there are multiple comparisons among different small training datasets, a multiple comparison based on one-way ANOVA is employed to compare multiple results in this paper, and Holm's method (Holm 1979) is used for p value adjustment with overall significance level of 0.05. Table 1 shows the means and standard deviations of F, and means of reall and precision of GP Gaussian-based edge detectors evolved by each single image from the six images. To easily read images, we use $T_a - T_f$ to represent the training images 42078, 106020, 23025, 68077, 23080 and 216053, respectively. As can be seen, the evolved detectors from T_a and T_b have low performance on the 100 BSD test images, however the evolved detectors from T_c and T_e present good detection performance (compared with the Sobel edge detector with $F = 0.48$, Martin et al. 2004). An interesting observation is that the low performing edge detectors have a low recall. The average recall of the evolved edge detectors from T_c or T_b is lower than 0.5. Most of the edge detectors from the six images have precision around 0.5.

Table 2 gives a statistical comparison between all edge detectors evolved by the six images respectively. Note that all comparisons are based on F values in this paper. For a pairwise comparison, the first group is from a setting in the first column, and the second group is from a setting in the first row. Here, $\uparrow$ indicates that the first group is significantly better than the second group; and $\downarrow$ indicates that the first group is significantly worse than the second group; otherwise, — indicates no significant difference between the first and second groups. Since there are no direction influences on the comparison of two groups, some redundant comparison results are removed from the table. From the comparison results, the six groups of results can be divided into three sets. The first set is the worst performing edge detectors, namely the edge detectors from images T_a and T_b which are significantly worse than the edge detectors from the other four images. The second set is the middle level for the edge detectors from T_d and T_f. These edge detectors are significantly better than the worst edge detectors from T_a and T_b, but significantly worse than the edge detectors from T_c and T_e. The third set is the best level for these edge detectors from T_c and T_e. In each level, there are no significant differences between the edge detectors from different images, except for the worst level.

Training images T_a and T_b only contain a single object, and the contrast between the object and the background is high. The edge information in the two images is not rich. For T_d and T_f, the edge information from the two images is richer than the edge information from T_a and T_b. The two images in the middle level contain different objects and different graylevel gaps between different regions. Training images T_c and T_e contain the richest edge information, compared with the other four images. In T_e, there is rich edge information from the building and the person. In T_c, the edges at the boundary of the person are different, such as the person's boundary between the body and the wall and between the leg and the step, and the boundary of the basket. It seems that images including different edges are good to evolve GP Gaussian-based edge detectors.

Table 1 Test performance F values (mean ± standard deviation, SD), and means of recall and precision of GP Gaussian-based edge detectors from each of the six images as training data respectively

Image	F (mean ± SD)	Recall	Precision
42078 (T_a)	0.4072 ± 0.0500	0.3213	0.5741
106020 (T_b)	0.4353 ± 0.0342	0.4634	0.4158
23025 (T_c)	0.5402 ± 0.0077	0.5851	0.5029
68077 (T_d)	0.5006 ± 0.0220	0.5431	0.4812
23080 (T_e)	0.5267 ± 0.0128	0.5913	0.4767
216053 (T_f)	0.5146 ± 0.0139	0.5130	0.5205

Table 2 One-way ANOVA (row vs column) for GP Gaussian-based edge detectors from the six images as training data, respectively

	T_b	T_c	T_d	T_e	T_f
T_a	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
T_b		$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
T_c			$\uparrow$	–	$\uparrow$
T_d				$\downarrow$	–
T_e					$\uparrow$

5.1 Two images as training data

From Table 1, it is known that T_a obtains the lowest performance of the evolved edge detectors, therefore, T_a is mainly used to combine with other images as training data. Four new training datasets (groups) are used, and they are training data C1 including T_a and T_d, C2 including T_e and T_f, C3 including T_a and T_f, and C4 including T_a and T_b.

Table 3 gives the means of F, recall and precision values of the GP Gaussian-based edge detectors evolved by training datasets C1, C2, C3 and C4. First of all, from the table, the means of F values of the evolved edge detectors from all training data with two images are higher than 0.5. Although images T_a and T_b get the worst performance when they are individually used to train edge detectors, the combination of the two images C4 obtains good performance. Second, two images as the training data can obtain high recall, such as C1 and C2. However, the evolved edge detectors from C3 and C4 have low recall, but high precision.

Table 4 gives a comparison between each training data group using two images and each training data group using a

Table 3 Test performance F values (mean ± standard deviation(SD)), and means of recall and precision of GP edge detectors from two images as training data, respectively

Pair	F (mean ± SD)	Recall	Precision
C1	0.5349 ± 0.0213	0.6112	0.4869
C2	0.5328 ± 0.0117	0.6087	0.4753
C3	0.5028 ± 0.0181	0.4651	0.5512
C4	0.5011 ± 0.0205	0.4622	0.5540

Table 4 One-way ANOVA (row vs column) for GP edge detectors from two images and single image as training data

	T_a	T_b	T_c	T_d	T_e	T_f
C1	↑	↑	–	↑	–	–
C2	↑	↑	–	↑	–	–
C3	↑	↑	↓	–	↓	–
C4	↑	↑	↓	–	↓	–
S_{rnd}	↑	↑	–	↑	–	–

single image. First, from the comparisons among all results from all single image training data and each training data with two images, the evolved edge detectors from C4 are significantly better than the evolved edge detectors from T_a and T_d when they are individually used to train edge detectors respectively. Second, the training data using an image from the worst level (of a single image as the training data) and an image from the middle level, namely C1 and C3, obtain results which are significantly better than the results from the worst level, but not significantly different from the results from the middle level. Third, the training data using an image from the best level and an image from the middle level, namely C2, obtain the results which are significantly better than the results from the worst level and the middle level. Fourth, the evolved edge detectors from all four training datasets using two images are significantly better than the two worst level individual images as training data respectively. From these comparisons, it seems that the combination of two images as training data can improve the performance of evolved detectors. When an image is randomly selected as training data, the evolved edge detectors might have low performance. If two images are randomly chosen as training data, the evolved edge detectors at least do not have too low performance. For a single image as training data, the comparison between S_{rnd} and a single fixed image will be discussed in Sect. 5.2.

Table 5 shows the comparisons among the different combinations based on two images. The table reveals that the evolved edge detectors from the combinations C1 and C2 are significantly better than the evolved edge detectors from the combinations C3 and C4. There are no significant differences between C1 and C2, and between C3 and C4. The combinations with one image from the worst level and one image from the middle level might have similar performance to the combination with both images from the middle level, and might be the same as the combination with both images from the worst level as well.

Table 5 One-way ANOVA (row vs column) for GP edge detectors from two images as training data respectively

	C2	C3	C4
C1	–	↑	↑
C2		↑	↑
C3			–

Table 6 F values (mean ± standard deviation(SD)), and means of recall and precision of GP edge detectors from sampling techniques on 20 images

Training	F (Mean ± SD)	Recall	Precision
S_{sub}	0.4940 ± 0.0260	0.8403	0.3517
S_{rnd}	0.5279 ± 0.0383	0.6989	0.4485
S_{ada}	0.5521 ± 0.0290	0.6716	0.4869
S_{20}	0.5628 ± 0.0131	0.6681	0.4893

Comparing the details of the four combinations of two images from the six images in Fig. 2, some interesting observations on edge characteristics in these images are found. First, in C1, although image T_a only has a very obvious object, the graylevels of the boat and hill in image T_d are close to the graylevels of background. Second, in C3, most of the graylevels of the building and person can be clearly distinguished from the background, which is similar to the difference between the object and background in image 42078. Last, high contrast exists in T_a and T_b (C4), but the edges at two different regions in T_b are totally different from the edges existing in T_a, which enriches edge information in C4. From the different combinations of two images, it seems that a combination with different edge information can improve the evolved edge detectors' performances.

5.2 Sampling

Table 6 gives the test results from the different sampling techniques and the training data using all 20 images. Comparing with the training data using a single image and using two images, it is found that recalls of the evolved edge detectors from different sampling techniques or 20 images are very high. Although S_{sub} has the highest recall average in the table, its precision average is the lowest. The lowest average of F values for evolved edge detectors is from S_{sub}. The highest average of F values for evolved edge detectors is from S_{20}, and the mean of F values of the edge detectors from S_{ada} is close to the mean of F values of the edge detectors from S_{20}.

Table 7 One-way ANOVA (row vs column) for GP Gaussian-based edge detectors from sampling techniques on 20 images

	S_{rnd}	S_{ada}	S_{20}
S_{sub}	↓	↓	↓
S_{rnd}		↓	↓
S_{ada}			–

Table 7 presents the comparison results among S_{sub}, S_{rnd}, S_{ada} and S_{20}. The evolved edge detectors from the sampling technique using subimages S_{sub} are significantly worse than the sampling techniques using original full images.

Although the evolved edge detectors from S_{rnd} are significantly worse than the evolved edge detectors from S_{ada} and S_{20}, the difference between the mean of F values from the sampling technique S_{rnd} and that of S_{20} is less than 0.04. Therefore, even though only one image is randomly selected as evaluation data at each generation, the varying training data can still lead to good edge detectors, compared with the training data using all 20 images. Also, from Table 4, the evolved edge detectors from S_{rnd} are significantly better than the evolved edge detectors from T_a, T_b and T_d, but not significantly different from the evolved edge detectors from the other three images in the table. It shows that a random image at a generation as the evaluation data is generally better than a fixed image in the whole evolution process as the evaluation data.

Compared with S_{sub} and S_{rnd}, S_{ada} obtains evolved edge detectors which are significantly better. The increasing number of images selected at each generation possibly makes GP select good Gaussian-based edge detector candidates at each generation. Since there are no significant differences between the evolved edge detectors from S_{ada} and S_{20}, the sampling technique with varying images in a set is efficient to train edge detectors which have similar performance to the edge detectors evolved with all images in the set. Since the number of training images is from four to eight, the average number of training images in all generations is six in the worst case. From the test performance for the edge detectors evolved by S_{ada}, it seems that only using several training images from a set can train edge detectors performing similarly to the edge detectors evolved by all images in the set. This sampling technique considerably reduces the computational cost over using the 20 images, while maintaining a similar performance.

5.3 GP edge detectors vs existing Gaussian-based edge detectors

In order to validate the performance of GP Gaussian-based edge detectors, the Gaussian gradient (GG) and the surround suppression (SS expressed by GG_{SS}) are selected as existing Gaussian-based edge detectors to compare with the evolved edge detectors. The Canny edge detector (Canny 1986) employs the Gaussian gradient in the feature extraction stage and non-maximum suppression and hysteresis thresholding in the post-processing stage. Since this paper mainly focuses on feature extraction, specific post-processing techniques, such as non-maximum suppression, are not considered. After filtering noise in the pre-processing stage and using non-maximum suppression and hysteresis thresholding in the post-processing stage, the Sobel and Prewitt edge detectors can be approximated to the Canny edge detector (Papari and Petkov 2011). Rather than comparing the final results from the Canny edge detector, we compare the GP edge detectors with the results from the Gaussian gradient (used in the Canny edge detector) and SS.

From the outputs of GG and SS, different thresholds are used to choose the maximum F on the BSD test images as test performance for GG and SS, and $F_{\max}$ is employed for their test performance. Based on 52 different thresholds $\frac{k}{52}$ $(k = 0, 1, \ldots, 51)$, it is found that the $F_{\max}$ for GG is 0.5153, and SS is 0.5381. Note that the edge responses from GG and SS are mapped to the range from 0 to 1. Here $\sigma \in \{1.0, 2.0, 5.0\}$ is used to choose the best $F_{\max}$ for GG, and for the suppression based on the DoG in SS, the best test performance is chosen from the three sets of the σ pairs (1.0, 2.0), (1.0, 3.0) and (0.8, 3.0). Note that $\sigma = 1.0$ is a very common setting (Basu 2002), and the other values are used for different detection results. The evolved Gaussian-based edge detectors are rotation invariant, so only the invariant version of surround suppression is chosen, and different directions are not considered.

Note that we only select the Gaussian gradient as an edge detector using a single Gaussian filter for the comparison. From the existing report (Arbeláez et al. 2011), the Sobel and Prewitt edge detectors have similar detection performance ($F = 0.48$), and the DoG edge detector has $F = 0.50$. Since GG ($F = 0.5153$) has higher detection performance than the three edge detectors, the other three edge detectors are not chosen for the comparison.

Table 8 shows the comparisons among GG, SS, image T_c, C1, S_{ada} and S_{20}. Here, the highest mean of F values is chosen from training data using a single image, two images, one of the three sampling techniques (S_{sub}, S_{rnd} and S_{ada}), and the 20 images. From the table, only a single Gaussian filter (GG) has the lowest performance in these compared results. In general, the evolved Gaussian-based edge detectors include several Gaussian filters, and SS contains three Gaussian filters. The evolved edge detectors from image T_c and C1 have no significant differences from SS, but the evolved edge detectors from S_{ada} and S_{20} are significantly better than SS.

In general, only using a single image as the training data is not sufficient to train edge detectors. However, the evolved edge detectors from the single training image T_c are not worse

Table 8 One-way ANOVA (row vs column) comparing F values from some GP edge detectors with $F_{\max}$ values from Gaussian gradients (GG) and surround suppression (SS) on the BSD test images

	SS	T_c	C1	S_{ada}	S_{20}
GG	↓	↓	↓	↓	↓
SS		–	–	↓	↓
T_c			–	–	↓
C1			–	↓	↓
S_{ada}					–

Table 9 Comparison between Gaussian-based GP edge detectors and GP edge detectors using blocks of pixels (GP$_{block}$) from Fu et al. (2012c)

	F	Recall	Precision
GP$_{block}$	0.5248 ± 0.0190	0.5705	0.4890
S_{ada}	0.5521 ± 0.0290 ↑	0.6716	0.4869
S_{20}	0.5628 ± 0.0131 ↑	0.6681	0.4893

than the other results in Table 8, except for the results from S_{20}. It seems that it is possible to use a single image as training data to evolve good edge detectors as long as it includes rich enough edge information.

The small set C1 is included in S_{20}, and the evolved edge detectors are significantly better than GG and SS, but these edge detectors from C1 are significantly worse than the evolved edge detectors from S_{ada} and S_{20}. It seems that two images as training data can be used to evolve good edge detectors. Also, using a set of images including the two images is better than using the two images to train Gaussian-based edge detectors, even when the evaluation data at each generation does not use all images in the set (the number of selected images is more than 3).

5.4 Gaussian-based GP edge detectors vs GP edge detectors using blocks of pixels

Table 9 shows the comparisons between the Gaussian-based GP edge detectors and the GP edge detectors using blocks of pixels from Fu et al. (2012c). Since the edge detectors evolved by GP using single raw pixels are significantly worse than the edge detectors evolved by GP using blocks of pixels (GP$_{block}$) Fu et al. (2012c), we only employ the GP edge detectors using blocks of pixels to compare with the Gaussian-based edge detectors. In Table 9, ↑ indicates that the Gaussian-based edge detectors are significantly better than GP$_{block}$ based on the t-tests with significance level 0.05, in terms of F. Although their detection precisions are similar, the Gaussian-based GP edge detectors have higher recall than the GP edge detectors using blocks of pixels. It seems that the Gaussian-based GP system, employing prior domain knowledge (Gaussian-based filtering techniques), has ability to improve detection performance, compared to the method of search blocks of pixels.

5.5 Detected visual results from the best GP edge detectors

Figure 3 shows five example images from the BSD test image dataset and their ground truth (GT). Also, the detected results from GG and SS are shown in Fig. 3. The results detected by GG and SS are based on their own best threshold in the overall view of the BSD 100 test images. The best threshold means that F is maximum when the threshold from the set of thresholds ($\frac{k}{52}$) is used. The overall best thresholds for GG and SS are 0.2793 and 0.0392, respectively. These selected images in Fig. 3 have different textures, and their edges are not easily detected by a normal Gaussian filter, except for image 42049, when the task is very easy.

To visually compare the detected results, Fig. 4 shows these four images detected by the best evolved edge detector from T_c ($F = 0.5531$), C1 ($F = 0.5699$), S_{sub} ($F = 0.5607$), S_{rnd} ($F = 0.5750$), S_{ada} ($F = 0.5998$) and S_{20} ($F = 0.5879$).

First, from the detected results for image 42049, all edge detectors give a good detection result. Only noise from the branch slightly affects the Gaussian-based edge detectors from T_c, S_{sub}, S_{ada} and S_{20} to perform detection. From the detected results on the 100 BSD test images, it seems that evolved Gaussian-based edge detectors can perform detection well on simple (non-texture) images.

Second, the detected results on the other four different images with textures show the different performance on these edge detectors. For the four images, GG is strongly affected by the different textures in these images. SS can reduce the influence from textures, but is still affected by some textures, such as the irregular background in image 175032 (sticks being out of order) and the heavy texture on the wall in image 385039. Also, SS fails to detect edges in low contrast, such as the body of the object in image 69020.

From the detected results by the evolved Gaussian-based edge detectors, the best edge detector from S_{sub} detects most of the edges, but it is strongly affected by textures, compared with the other evolved edge detectors. The best evolved edge detector from T_c detects images 69020 and 385039 with very slight influence from textures, but it is still hard to detect the top boundary of the object in image 69020. For image 175032, it is strongly affected by the irregular background, compared to the detection from C1, S_{rnd}, S_{ada} and S_{20}. The best evolved edge detector from C1 has similar detection for image 69020 to the edge detector from T_c, but it has stronger influence from the wall in image 386039 and weaker influence from the irregular background in image 175032, compared with the latter. The best edge detector from S_{ada} is not affected by the animal skin texture in image 87046, whereas the other edge detectors are heavily affected

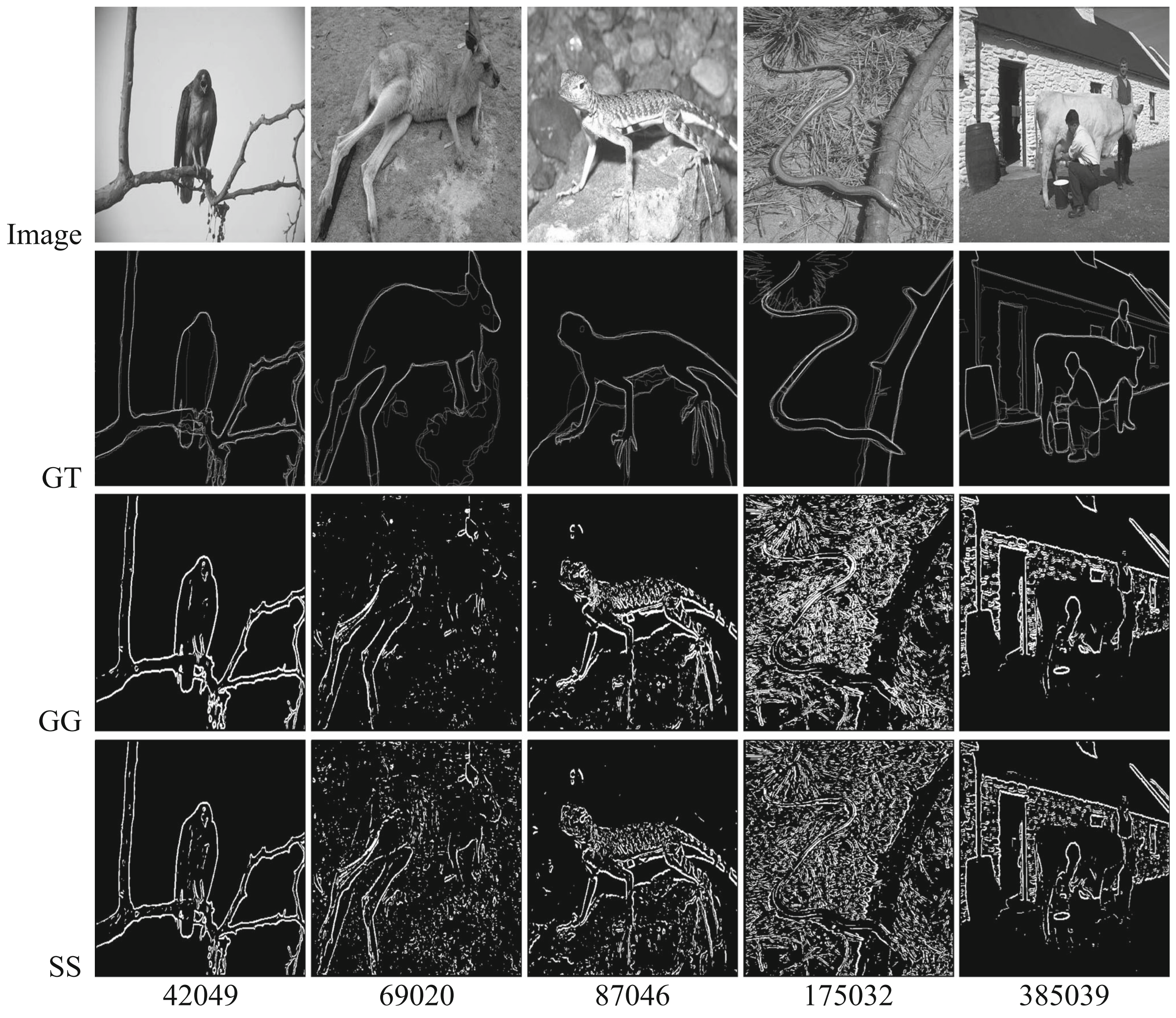

Fig. 3 Five example BSD test images detected by Gaussian gradient (GG) and surround suppression (SS)

by it. Note that some of the evolved Gaussian-based edge detectors have clear responses on the boundaries among the stones in the background, but are not as affected by the skin texture as GG and SS. Also, the edge detector from S_{ada} has good detection results for the other images. Based on the BSD test images, the edge detector from S_{ada} is the best edge detector among all evolved Gaussian-based edge detectors in Fig. 4. From the visually detected results, the GP Gaussian-based edge detectors have good ability to suppress textures.

A potential reason for GP evolving good Gaussian-based edge detectors is that the GP system automatically constructs composite filters with different types of Gaussian filters based on different parameters. Since the tree-based GP system has flexible presentations, there is a large space for combining Gaussian filters. Note that each Gaussian filter automatically generated in GP has some ability to extract edges, so almost all combinations of Gaussian filters from GP have some ability to extract edges. The GP system almost only focuses on the searching for good combinations of Gaussian filters.

5.6 Example evolved GP Gaussian-based edge detector

In order to directly present a GP Gaussian-based edge detector, an evolved program is selected from using T_c, which is shown in Fig. 5, where “C” is combination function $\mathbb{C}$. Equation (11) describes this GP Gaussian-based edge detector GGP.

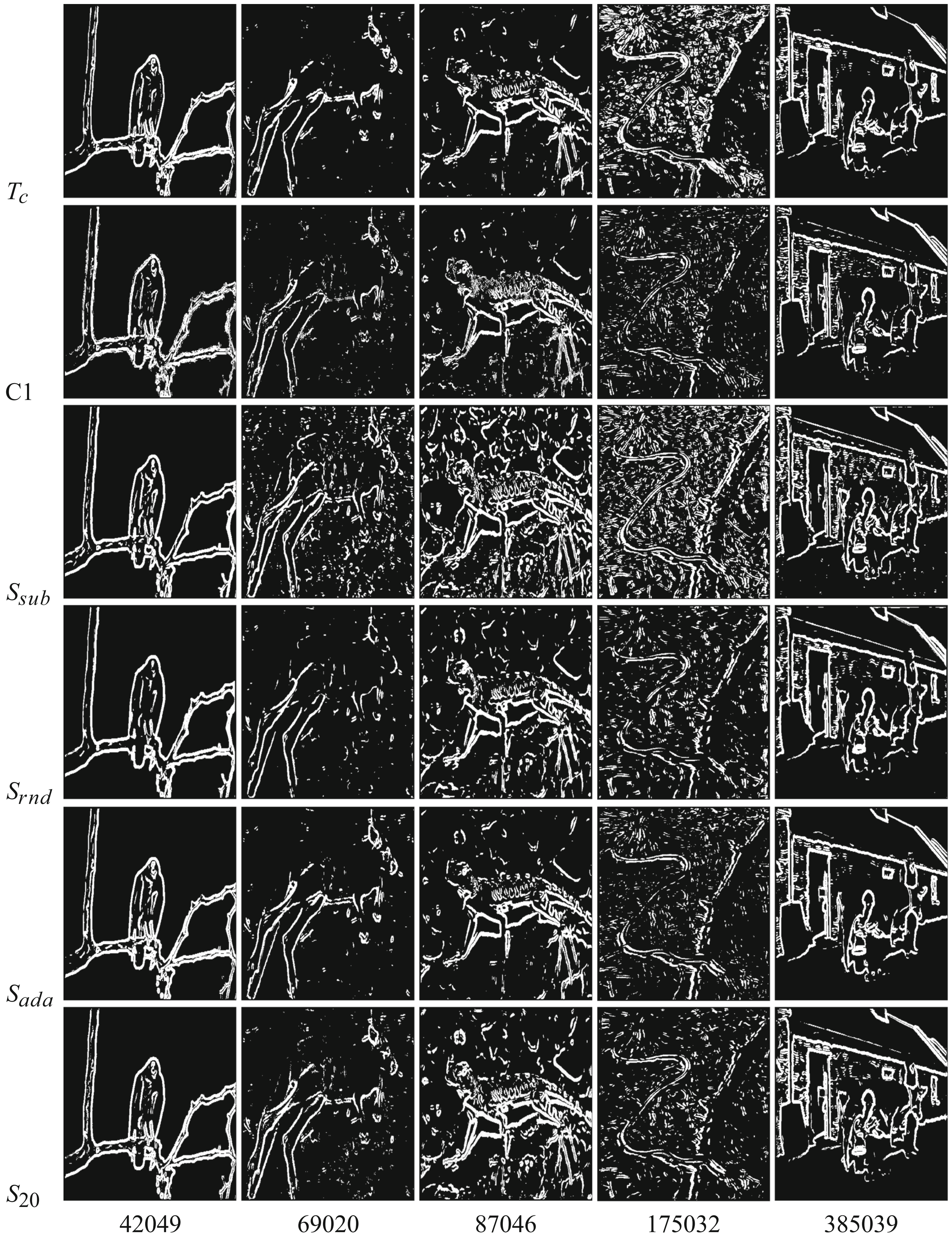

Fig. 4 Five example BSD test images detected by the GP Gaussian-based edge detectors

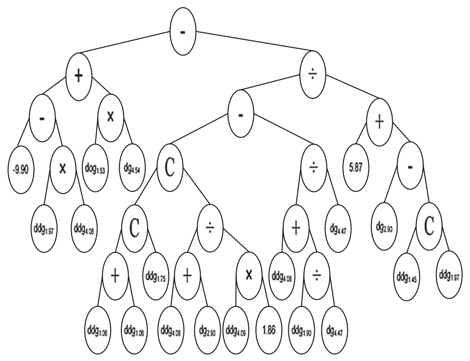

Fig. 5 One example GP Gaussian-based edge detector (GGP)

$$GGP = dog_{1.53} \times dg_{4.55} - ddg_{1.97} * ddg_{4.08} - 9.90 - \frac{2ddg_{1.06}\mathbb{C}ddg_{1.75}\mathbb{C}\frac{ddg_{4.08}+dg_{2.90}}{1.86ddg_{4.09}} - \frac{\frac{ddg_{1.90}}{dg_{4.47}}+ddg_{4.08}}{dg_{4.47}}}{5.87 + dg_{2.90} - ddg_{1.45}\mathbb{C}ddg_{1.97}} \quad (11)$$

From this equation, it can be seen that the evolved solution includes different combinations of Gaussian filters, namely multiplication, division, convolution and difference of different combinations. For the parameter σ, GGP includes both fine and coarse solutions for detecting images. Also, this solution includes the three types of Gaussian filters from the terminal set. It seems that the combination of using the three types of Gaussian filters is good to detect edges, rather than using only one alone. Note that $\mathbb{C}$ can be used to suppress texture gradients. For example, "$dg_{2.90} - ddg_{1.45}\mathbb{C}ddg_{1.97}$" is similar to the suppression technique. The division might be potentially used to suppress noise, and the difference of different combinations and the multiplication possibly enhance the response from the true edges. However, the edge detector is very complicated; further experiments and analyses are required in the future.

Additionally, it is hard to analyse the influence of each parameter because of the complicated structure of GGP. We have investigated soft edge maps in GP low-level edge detectors (Fu et al. 2012e), and there might be two problems from the outputs of GP edge detectors before binarization. The first problem is that the outputs of a GP edge detector (before binarization) spread largely. The second problem is that some of outputs are very closed to the fixed threshold used in GP. When GGP is applied to a BSD test image, the outputs of GGP (before binarization) spread largely. For example, to test image 101085, the maximum output value is 1.37×10^9 and the minimum output value is -2.69×10^{33}. When the linear transformation from Fu et al. (2012e) is used, most of pixels are mapped to the highest scale level. In future work, we will get suitable soft edge maps from GP Gaussian-based edge detectors based on our previous work (Fu et al. 2012e), then analyse the parameters in GP Gaussian-based edge detectors.

6 Further discussions

This section discusses the computational cost, thickness of detected edges existing in GP Gaussian-based edge detectors, and contributions of Gaussian filters, and the combination function $\mathbb{C}$.

6.1 Computational cost

The time for a GP run evolving a Gaussian-based edge detector with 200 generations is around 4.5 days on a single machine with CPU 3.1 GHz when S_{20} (20 images) is used as the training data. So this is a time-consuming algorithm, but this can be sped up by a computational grid. There are two reasons for the heavy computational cost. First, the maximum tree depth for the GP Gaussian-based programs is seven, so it is possible that a GP program includes more than ten Gaussian filters. In the training stage, the weights of a Gaussian filter in a new program need to be calculated. The time for calculating responses on all Gaussian filters in a program will take most of the execution time. Second, the training data have 3088020 ($20 \times 481 \times 321$) pixels, and a GP program needs to take some time to discriminate all pixels.

However, the training time for only using a single image as the training data is around five and half hours in a run. When the training data uses two images, the training time for each run is about 11 h. For the sampling techniques, the training time for using S_{rnd} as the training data is very close to the training time for using a single image as the training data. It costs about 33 h to use S_{ada} to evolve a GP Gaussian-based edge detector. The sampling technique S_{ada} reduces more than two-thirds of the training time cost in S_{20}. Therefore, efficiently selecting images can remarkably reduce the computational cost in the GP system, while maintaining good performance.

All GP evolved edge detectors only take less than half a second to detect a BSD image (in the test stage). The depth of the solution in Fig. 5 is seven (the maximum depth), and the solution includes 17 Gaussian filters and three function operators $\mathbb{C}$, which makes the solution detect one BSD image in slightly longer than 0.1 s. However, simplification of the evolved detector can make this detection time even shorter. From Eq. (11), the Gaussian filters $ddg_{1.06}$ and $dg_{4.47}$ can be only calculated once, not twice in the tree-based program in Fig. 5. It is possible to replace $ddg_{1.97}$, $ddg_{1.75}$ and $ddg_{1.45}$ with an approximate LoG. Also, $ddg_{4.08}$ and $ddg_{4.09}$ might be replaced by another approximate LoG. How to automatically and efficiently simplify evolved GP Gaussian-based edge detectors would be investigated in the future. In addition, a restriction on the number of Gaussian filters in a GP edge detector needs to be investigated so that an evolved GP edge detector can perform edge detection in less than 0.1 s.

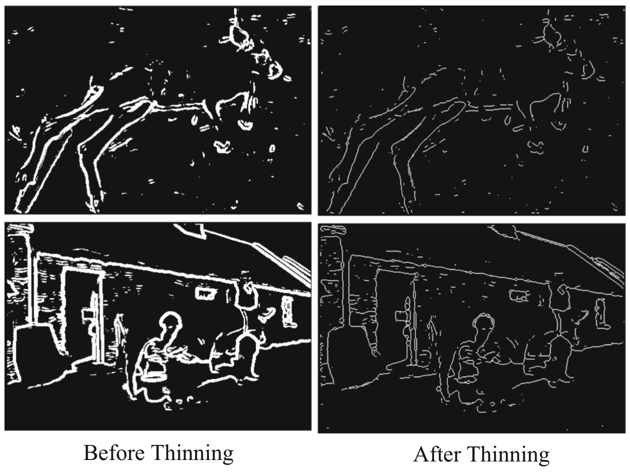

Fig. 6 Two example images detected by a Gaussian-based edge detector before and after thinning operations (Lam et al. 1992)

Table 10 Results (F values, mean ± standard deviation, SD) of evolved edge detectors from single type of filter without combination function $\mathbb{C}$ (with T_c) on the BSD test images

Setting	Mean ± SD	Max	Min	p value
dg_σ	0.5258 ± 0.0029	0.5308	0.5206	0.0000 ↓
dog_σ	0.5004 ± 0.0074	0.5208	0.4809	0.0000 ↓
ddg_σ	0.4921 ± 0.0046	0.5004	0.4823	0.0000 ↓

Table 11 One-way ANOVA (row vs column) comparisons (F) among single Gaussian filters used in GP on the BSD test images

	dog_σ	ddg_σ
dg_σ	↑	↑
dog_σ		↑

6.2 Thickness of detected edges

The edges detected by the GP edge detectors (see Fig. 3) often have thick responses on edges. A potential reason is that large scale Gaussian filters exist in the GP edge detectors. For instance, Eq. (11) includes $ddg_{4.08}$, $ddg_{4.09}$ and $dg_{4.47}$. Another potential reason is that the ground truth images in the training data (see Fig. 2) are combined with different observations, so the width of a true edge is possibly more than one pixel. The overlap in prediction for a true edge point is allowed to have more than one response in the fitness function FOM. Figure 6 shows two example detected edge maps from S_{ada} in Fig. 3 simply thinned by a thinning operation (Lam et al. 1992). Non-maximum suppression (Canny 1986) could easily be used to thin the detection results, which is not the focus of this paper, but can be investigated in the future.

6.3 Contributions on different Gaussian filters

Influence of using different types of Gaussian filters is now investigated. In order to check whether using different types of Gaussian filters is better than using a single type of Gaussian filter only, the terminal set now only chooses a single type of filter to evolve programs. The combination function $\mathbb{C}$ might influence the performance of evolved edge detectors, so $\mathbb{C}$ is removed from the function set. The influence of $\mathbb{C}$ will be discussed in the next subsection. Since T_c can be used to obtain good evolved edge detectors, the single image is employed as the training data for analysis of using different types of filters to evolve edge detectors.

Table 10 shows the results from only using dg_σ, dog_σ and ddg_σ respectively. Here, the best evolved edge detector (Max) and the worst evolved edge detector (Min) for using each single type of filter are given. The p values are obtained by the two-sample t-tests between the relevant results and the results using all functions when only T_c is used as the training data.

From the tests (p values), only using a single type of filter is not sufficient to evolve good edge detectors. All of the results from using a single type of filter obviously decrease the test performance, and they are significantly worse than using the three types of filters. Therefore, only combining the same type of Gaussian filter is not as good as combining different types of Gaussian filters for extracting edge features, which is similar to the surround suppression technique using different types of filters to improve detection performance.

Table 11 reveals the comparison among the results from the three types of filters. It is interesting that the results from only using dog_σ are significantly better than the results from only using ddg_σ, although dog_σ is considered as approximation of ddg_σ (Basu 2002). A potential reason is that each dog_σ filter includes two different Gaussian filters, but a ddg_σ filter only uses a single Gaussian filter. After combining filters with different parameters, the responses from the combination of dog_σ are richer than the responses from the combination of ddg_σ. Therefore, it is possible to obtain better results from only using dog_σ than only using ddg_σ.

Also, the results from using dg_σ filters are significantly better than the results from using dog_σ and ddg_σ, respectively. The mean of F values from using dg_σ filters is 0.0144 less than the mean of F values from using all functions in the GP system (see Table 1) when T_c is used. It seems that the image Gaussian gradient is important for extracting edge features. However, only combining dg_σ filters is not sufficient to improve detection performance. From the comparison among the three types of filters, it seems that dog_σ and ddg_σ help dg_σ to improve detection performance. In surround suppression, the major responses come from the image gradient. The GP Gaussian-based edge detectors also show similar behaviour.

Table 12 Results (F values, mean ± standard deviation, SD) of evolved edge detectors by using single filters and combination function $\mathbb{C}$ (with T_c) on the BSD test images

Setting	Mean ± SD	Max	Min	p value
$\{dg_\sigma, \mathbb{C}\}$	0.5156 ± 0.0135	0.5379	0.4789	0.0002 ↓
$\{dog_\sigma, \mathbb{C}\}$	0.4856 ± 0.0334	0.5355	0.4203	0.0247 ↓
$\{ddg_\sigma, \mathbb{C}\}$	0.4786 ± 0.0376	0.5387	0.3512	0.0592

Table 13 One-way ANOVA (row vs column) comparisons (F) among single Gaussian filters used in GP (including function $\mathbb{C}$) on the BSD test images

	$\{dog_\sigma, \mathbb{C}\}$	$\{ddg_\sigma, \mathbb{C}\}$
$\{dg_\sigma, \mathbb{C}\}$	↑	↑
$\{dog_\sigma, \mathbb{C}\}$		–

6.4 Combination function $\mathbb{C}$

The combination function $\mathbb{C}$ is not used for the experiments in Table 10. It is possible that the difference between using a single type of filter and the results using the three types of filters are affected by the function $\mathbb{C}$. In order to investigate the influence from function $\mathbb{C}$, the function is added into each single type of filter to evolve edge detectors. Table 12 shows the results for the three single types of filters after using function $\mathbb{C}$. Here the p values are obtained by the two-sample t-tests between without function $\mathbb{C}$ and using function $\mathbb{C}$. It is surprising that the test performances from respectively using dg_σ and dog_σ are decreased after using function $\mathbb{C}$.

There are some interesting observations in Table 12. First, using each type of filter combined with function $\mathbb{C}$ decreases the detection performance of the evolved edge detectors, in terms of the mean of F values. Second, the standard deviations of the evolved edge detectors become larger after using function $\mathbb{C}$. This indicates that the performance of evolved edge detectors becomes less stable. Third, from the comparisons of the best edge detectors and the worst edge detectors in each type of filter with and without function $\mathbb{C}$, the range of the performance of the evolved edge detectors after using function $\mathbb{C}$ becomes large. The best evolved edge detector from each setting with function $\mathbb{C}$ has a higher F value than the best edge detector without function $\mathbb{C}$. However, the worst edge detector from each setting in Table 12 is obviously worse than the worst edge detector from the relevant setting in Table 10. In particular, there is very obvious influence on ddg_σ after using function $\mathbb{C}$, in terms of the performance on the best and worst evolved edge detectors. There is no significant difference between the results from using and without function $\mathbb{C}$ when only ddg_σ is used. However, the relevant worst evolved edge detector (in $\{ddg_\sigma, \mathbb{C}\}$) has very low F, and the relevant best evolved edge detector has a higher F than the best evolved edge detector from ddg_σ in Table 10.

From the three interesting observations, the function $\mathbb{C}$ certainly influences the evolved edge detectors. It seems that using $\mathbb{C}$ might increase the ability to find good edge detectors from an overview. However, after adding function $\mathbb{C}$ in the function set, the worst evolved edge detector in each setting becomes worse. There are at least two possible reasons for this phenomenon. The first one is that good combinations of a single type of filter and function $\mathbb{C}$ are very hard to find. The other potential reason is that overfitting occurs when only a single type of filter and function $\mathbb{C}$ are employed in the GP system.

In addition, the best edge detector for each setting in Table 12 has a lower F than the mean of the evolved edge detectors from Table 1 when T_c is used. It also shows that using a single type of filter is not good to evolve Gaussian-based edge detectors.

Table 13 gives the comparisons among only using a single type of filter with function $\mathbb{C}$. It also shows that the Gaussian gradient is better than the Gaussian second-order derivative to extract edge features. When adding function $\mathbb{C}$ in the function set, there is no significant difference between only using dog_σ and only using ddg_σ.

7 Conclusions

The goal of this paper was to investigate evolving Gaussian-based edge detectors using GP. From the different training data, the goal was successfully achieved by using a proposed Gaussian-based GP system to evolve edge detectors. From the results, the GP evolved programs performed better than the Gaussian gradient and the invariant surround suppression. A single image with rich edge information *could* be used to evolve good edge detectors. The proposed sampling technique of choosing several images from a set of images at each generation was effectively used for evolving edge detectors. Therefore, a small set of images has potential to evolve good Gaussian-based programs. In addition, from the investigation on each type of filter for constructing Gaussian-based edge detectors, the Gaussian gradient should be the main filter in a combination; the combination function $\mathbb{C}$ should choose different types of Gaussian filters as its inputs so that edge detectors with good generalisation ability could be obtained.

In terms of running time, the methods proposed in this paper require a relatively long evolutionary training time, but the evolved detectors can detect edges in new/unseen images in a very short time. Accordingly, this approach can be used to detect edges in the scenarios that allow a long learning time.

For future work, we will focus on the following three aspects: (1) the estimated distribution for the characteristics in different images so that a GP system can automatically choose images to efficiently train Gaussian-based edge detectors; (2) investigation of a dynamic threshold technique for the GP system so that the system mainly focuses on the selection of good structures; and (3) the restriction of the number of Gaussian filters in a GP edge detector so that the GP evolved edge detector can rapidly give detection results.

References

Arbeláez P, Maire M, Fowlkes C, Malik J (2011) Contour detection and hierarchical image segmentation. IEEE Trans Pattern Anal Mach Intell 33(5):898–916

Basu M (2002) Gaussian-based edge-detection methods: a survey. IEEE Trans Syst Man Cybern Part C Appl Rev 32(3):252–260

Bennamoun M, Boashash B, Koo J (1995) Optimal parameters for edge detection. Proc IEEE Int Conf Syst Man Cybern 2:1482–1488

Bergholm F (1987) Edge focusing. IEEE Trans Image Process 9:726–741

Bolis E, Zerbi C, Collet P, Louchet J, Lutton E (2001) A GP artificial ant for image processing: preliminary experiments with EASEA. In: Proceedings of the 4th European conference on genetic programming, pp 246–255

Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 8(6):679–698

Dollar P, Tu Z, Belongie S (2006) Supervised learning of edges and object boundaries. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 2:1964–1971

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

Espejo PG, Ventura S, Herrera F (2010) A survey on the application of genetic programming to classification. IEEE Trans Syst Man Cybern Part C Appl Rev 40:121–144

Fink M (2004) Object classification from a single example utilizing class relevance metrics. In: Proceedings of the neural information processing systems

Fu W, Johnston M, Zhang M (2011) 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 (2012a) 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 (2012b) Genetic programming for edge detection based on figure of merit. In: Proceedings of the genetic and evolutionary computation conference, pp 1483–1484

Fu W, Johnston M, Zhang M (2012c) 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 (2012d) 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 (2012e) Soft edge maps from edge detectors evolved by genetic programming. In: Proceedings of the IEEE congress on evolutionary computation, pp 24–31

Fu W, Johnston M, Zhang M (2013a) Automatic construction of gaussian-based edge detectors using genetic programming. In: Proceedings of the European conference on applications of evolutionary computation, pp 365–375

Fu W, Johnston M, Zhang M (2013b) 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 (2013c) 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 Trans Syst Man Cybern Part B Cybern 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 (2003) Contour detection based on nonclassical receptive field inhibition. IEEE Trans Image Process 12(7):729–739

Grigorescu C, Petkov N, Westenberg MA (2004) Contour and boundary detection improved by surround suppression of texture edges. Image Vis Comput 22(8):609–622

Harding S, Banzhaf W (2008) Genetic programming on GPUs for image processing. Int J High Perform Syst Architect 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

Holm S (1979) A simple sequentially rejective multiple test procedure. Scand J Stat 6(2):65–70

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

Kokkinos I (2010) Boundary detection using F-measure-, filter- and feature- (F3) boost. In: Proceedings of the 11th European conference on computer vision: part II, pp 650–663

Kunt M (1982) Edge detection: a tutorial review. Proc IEEE Int Conf Acoust Speech Signal Process 7:1172–1175

Lacroix V (1990) The primary raster: a multiresolution image description. In: Proceedings of the 10th international conference on pattern recognition, vol I, pp 903–907

Lam L, Lee SW, Suen C (1992) Thinning methodologies—a comprehensive survey. IEEE Trans Pattern Anal Mach Intell 14(9):869–885

Li FF, Fergus R, Perona P (2006) One-shot learning of object categories. IEEE Trans Pattern Anal Mach Intell 28(4):594–611

Lindeberg T (1996) Edge detection and ridge detection with automatic scale selection. In: Proceedings of 1996 IEEE computer society conference on computer vision and pattern recognition, pp 465–470

Marr D, Hildreth E (1980) Theory of edge detection. Proc R Soc Lond Ser B Biol Sci 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 Trans Pattern Anal Mach Intell 26(5):530–549

Miller E, Matsakis N, Viola P (2000) Learning from one example through shared densities on transforms. Proc IEEE Conf Comput Vis Pattern Recognit 1:464–471

Papari G, Petkov N (2011) Edge and line oriented contour detection: state of the art. Image Vis Comput 29:79–103

Poli R (1996) Genetic programming for image analysis. In: Proceedings of the first annual conference on genetic programming, pp 363–368

Quintana MI, Poli R, Claridge E (2006) Morphological algorithm design for binary images using genetic programming. Genet Program Evol Mach 7:81–102

Schunck B (1987) Edge detection with Gaussian filters at multiple scales. In: Proceedings of the IEEE workshop on computer vision, representation and control, pp 208–210

Song DM, Li B (1998) Derivative computation by multiscale filters. Image Vis Comput 16(1):43–53

Song W, Feng G, Tiecheng L (2006) Evaluating edge detection through boundary detection. EURASIP J Appl Signal Process 2006:1–15

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 PI (2005) Evolving optimal feature extraction using multi-objective genetic programming: a methodology and preliminary study on edge detection. In: Proceedings of the genetic and evolutionary computation conference, pp 795–802