Available online at www.sciencedirect.com

SCIENCE DIRECT®

Applied Soft Computing 4 (2004) 175–201

Applied Soft Computing

www.elsevier.com/locate/asoc

# Object detection in multi-modal images using genetic programming

Bir Bhanu*, Yingqiang Lin

*Center for Research in Intelligent Systems, College of Engineering, University of California, Riverside, CA 92521, USA*

Received 8 March 2003; accepted 8 January 2004

**Abstract**

In this paper, we learn to discover composite operators and features that are synthesized from combinations of primitive image processing operations for object detection. Our approach is based on genetic programming (GP). The motivation for using GP-based learning is that we hope to automate the design of object detection system by automatically synthesizing object detection procedures from primitive operations and primitive features. There are many basic operations that can operate on images and the ways of combining these primitive operations to perform meaningful processing for object detection are almost infinite. The human expert, limited by experience, knowledge and time, can only try a very small number of conventional combinations. Genetic programming, on the other hand, attempts many unconventional combinations that may never be imagined by human experts. In some cases, these unconventional combinations yield exceptionally good results. To improve the efficiency of GP, we propose soft composite operator size limit to control the code-bloat problem while at the same time avoid severe restriction on the GP search. Our experiments, which are performed on selected regions of images to improve training efficiency, show that GP can synthesize effective composite operators consisting of pre-designed primitive operators and primitive features to effectively detect objects in images and the learned composite operators can be applied to the whole training image and other similar testing images.
© 2004 Elsevier B.V. All rights reserved.

*Keywords:* Object detection; Genetic programming; Composite feature; ROI extraction

## 1. Introduction

In recent years, with the advent of newer, much improved and inexpensive imaging technologies and the rapid expansion of the Internet, more and more images are becoming available. Recent developments in image collection platforms produce far more imagery than the declining ranks of image analysts are capable of handling due to the speed limitation of human beings in analyzing images. Relying entirely on human image experts to perform image processing, image analysis and image classification becomes more and more unrealistic. Building automatic object detection and recognition systems to take advantage of the speed of computer is a viable and important solution to the increasing need of processing a large quantity of images efficiently.

Designing automatic object detection and recognition systems is one of the important research areas

* Corresponding author. Tel.: +1-909-787-3954; fax: +1-909-787-3188.
*E-mail addresses:* bhanu@vislab.ucr.edu (B. Bhanu), yqlin@vislab.ucr.edu (Y. Lin).

1568-4946/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2004.01.004

in computer vision and pattern recognition [1,2]. The major task of object detection is to locate and extract regions of an image that may contain potential objects so that the other parts of the image can be ignored. It is an intermediate step to object recognition. The regions extracted during detection are called regions-of-interest (ROIs). ROI extraction is very important in object recognition, since the size of an image is usually large, leading to the heavy computational burden of processing the whole image. By extracting ROIs, the recognition system can focus on the extracted regions that may contain potential objects and this can be very helpful in improving the recognition rate. Also by extracting ROIs, the computational cost of object recognition is greatly reduced, thus improving the recognition speed. This advantage is particularly important for real-time applications, where the recognition accuracy and speed are of prime importance.

However, the quality of object detection is dependent on the type and quality of features extracted from an image. There are many features that can be extracted. The question is what are the appropriate features or how to synthesize features, particularly useful for detection, from the primitive features extracted from images. The answer to these questions is largely dependent on the intuitive instinct, knowledge, previous experience and even the bias of algorithm designers and experts in object detection and recognition.

In this paper, we use genetic programming (GP) to synthesize composite features, which are the output of composite operators, to perform object detection. A composite operator consists of primitive operators and it can be viewed as a way of combining primitive operations on images. The basic approach is to apply a composite operator on the original image or primitive feature images generated from the original one; then, the output image of the composite operator, called composite feature image, is segmented to obtain a binary image or mask; finally, the binary mask is used to extract the region containing the object from the original image. The individuals in our GP-based learning are composite operators represented by binary trees whose internal nodes represent the pre-specified primitive operators and the leaf nodes represent the original image or the primitive feature images. The primitive feature images are pre-defined, and they are not the output of the pre-specified primitive operators.

## 2. Motivation and related research

### 2.1. Motivation

In most imaging applications, human experts design an approach to detect potential objects in images. The approach can often be dissected into some primitive operations on the original image or a set of related feature images obtained from the original one. It is the expert who, relying on his/her rich experience, figures out a smart way to combine these primitive operations to achieve good detection results. The task of synthesizing a good approach is equivalent to finding a good point in the space of *composite operators* formed by the combination of primitive operators.

Unfortunately, the ways of combining primitive operators are infinite. The human expert can only try a very limited number of conventional combinations. However, a GP may try many unconventional ways of combining primitive operations that may never be imagined by a human expert. Although these unconventional combinations are very difficult, if not impossible, to be explained by domain experts, in some cases, it is these unconventional combinations that yield exceptionally good results. The incomprehensibility of some effective solutions learned by GP demonstrates the value of GP in the generation of new features for object detection, since human experts never figure out solutions incomprehensible to them. The inherent parallelism of GP and the high speed of current computers allow the portion of the search space explored by GP to be much larger than that by human experts. The search performed by GP is not a random search. It is guided by the fitness of composite operators in the population. As the search proceeds, GP gradually shifts the population to the portion of the space containing good composite operators.

### 2.2. Related research and contributions

Genetic programming, an extension of genetic algorithm, was first proposed by Koza [3] and has been used in image processing, object detection and object recognition. Harris and Buxton [4] applied GP to the production of high performance edge detectors for one-dimensional signals and image profiles. The method is also extended to the development of practical edge detectors for use in image processing and machine vision. Poli [5] used GP to develop effective image filters to enhance and detect features of interest or to build pixel classification-based segmentation algorithms. Bhanu and Lin [6] used GP to learn composite operators for object detection. Their experimental results showed that GP is a viable way of synthesizing composite operators from primitive operations for object detection. Stanhope and Daida [7] used GP to generate rules for target/clutter classification and rules for the identification of objects. To perform these tasks, previously defined feature sets are generated on various images and GP is used to select relevant features and methods for analyzing these features. Howard et al. [8] applied GP to automatic detection of ships in low-resolution SAR imagery by evolving detectors. Roberts and Howard [9] used GP to develop automatic object detectors in infrared images. Tackett [10] applied GP to the development of a processing tree for the classification of features extracted from images.

Belpaeme [11] investigated the possibility of evolving feature detectors under selective pressure. His experimental results showed that it is possible for GP to construct visual functionality based on primitive image processing functions inspired by visual behavior observed in mammals. The inputs for the feature detectors are images. Koeppen and Nickolay [12] presented a special two-dimensional texture filtering framework, based on the so-called 2D-Lookup, and its configuration by means of evolutionary algorithms. With its configuration evolved by GP, the 2D-Lookup framework allowed to represent a very large number of texture filters. Their experimental results demonstrated that although the framework may never find the globally optimal texture filters, it evolves the initialized solutions toward better ones. Johnson et al. [13] described a way of automatically evolving visual routines for simple tasks by using genetic programming. The visual routine models used in their work were initially proposed by Ullman [14] to describe a set of primitive routines that can be applied to find spatial relations between objects in an input image. Ullman proposed, that given a specific task, the visual routine processor compiled and organized an appropriate set of visual routines and applied it to a base representation. But as Johnson et al. [13] pointed out, Ullman did not explain how routines were developed, stored, chosen and applied. In their work, Johnson et al. applied typed genetic programming to the problem of creating visual routines for the simple task of locating the left and right hands in a silhouette image of a person. In their GP, crossover was performed by exchanging between two parents the subtrees of the same root return type. To avoid the code-bloat problem of GP, they simply canceled a particular crossover if it would produce an offspring deeper than the maximum allowable depth.

Unlike the work of Stanhope and Daida [7], Howard et al. [8] and Roberts and Howard [9], the input and output of each node of a tree in our system are images, not real numbers. When the data from node to node is an image, the node can contain any primitive operation on images. Such image operations do not make sense when the data is a real number. In our system, the data to be processed are images, and image operations can be applied to primitive feature images and any other intermediate images to achieve object detection results. In [7–9], image operations can only be applied to the original image to generate primitive feature images. Also, the primitive features defined in this paper are more general and easier to compute than those used in [7,8]. Unlike our previous work [6], we take off the hard size limit of composite operators and use a soft size limit to let GP search more freely while at the same time prevent the code-bloat problem. The training in this paper is not performed on a whole image, but on the selected regions of an image and this is very helpful in reducing the training time. Of course, training regions must be carefully selected and represent the characteristics of training images [15]. Also, two other types of mutation are added to further increase the diversity of the population. Finally, more primitive feature images are employed. The primitive operators and primitive features designed in this paper are very basic and domain-independent, not specific to a kind of imagery. Thus, our system and methodology can be applied to a wide variety of images. We show results using synthetic aperture radar (SAR), infrared (IR) and color video images.

## 3. Technical approach

In our GP-based approach, individuals are composite operators represented by binary trees. The search space of GP is the space of all possible composite operators. The space is huge, although there could be equivalent composite operators in terms of their output images. In the computer system, a pixel of an image can assume only finite values, the number of possible images is finite, but this number is huge and astronomical. Also, if we set a maximum composite operator size, the number of composite operators is also finite, but again this number is also huge and astronomical. To illustrate this, consider only a special kind of binary tree, where each tree has exactly 30 internal nodes and 1 leaf node and each internal node has only one child. For 17 primitive operators and only 1 primitive feature image, the total number of such trees is $17^{30}$. It is extremely difficult to find good composite operators from this vast space unless one has a smart search strategy.

### 3.1. *Design considerations*

There are five major design considerations, which involve determining the set of terminals, the set of primitive operators, the fitness measure, the parameters for controlling the evolutionary run, and the criterion for terminating a run.

#### 3.1.1. *The set of terminals*

The set of terminals used in this paper are 16 primitive feature images generated from the original image: the first one is the original image; the others are mean, deviation, maximum, minimum and median images obtained by applying templates of sizes $3 \times 3$, $5 \times 5$ and $7 \times 7$, as shown in Table 1. These images are the input to composite operators. GP determines which operations are applied on them and how to combine the results. To get the mean image, we translate a template across the original image and use the average pixel value of the pixels covered by the template to replace the pixel value of the pixel covered by the central cell of the template. To get the deviation image, we just compute the pixel value difference between the pixel in the original image and its corresponding pixel in the mean image. To get maximum, minimum and median images, we translate the template across the original image and use the maximum, minimum and median pixel values of the pixels covered by the template to replace the pixel value of the pixel covered by the central cell of the template, respectively.

Table 1
Sixteen primitive feature images used as the set of terminals

| No. | Primitive feature image | Description |
|---|---|---|
| 0 | PFIM0 | Original image |
| 1 | PFIM1 | $3 \times 3$ Mean image |
| 2 | PFIM2 | $5 \times 5$ Mean image |
| 3 | PFIM3 | $7 \times 7$ Mean image |
| 4 | PFIM4 | $3 \times 3$ Deviation image |
| 5 | PFIM5 | $5 \times 5$ Deviation image |
| 6 | PFIM6 | $7 \times 7$ Deviation image |
| 7 | PFIM7 | $3 \times 3$ Maximum image |
| 8 | PFIM8 | $5 \times 5$ Maximum image |
| 9 | PFIM9 | $7 \times 7$ Maximum image |
| 10 | PFIM10 | $3 \times 3$ Minimum image |
| 11 | PFIM11 | $5 \times 5$ Minimum image |
| 12 | PFIM12 | $7 \times 7$ Minimum image |
| 13 | PFIM13 | $3 \times 3$ Median image |
| 14 | PFIM14 | $5 \times 5$ Median image |
| 15 | PFIM15 | $7 \times 7$ Median image |

#### 3.1.2. *The set of primitive operators*

A primitive operator takes one or two input images, performs a primitive operation on them and stores the result in a resultant image. Currently, 17 primitive operators are used by GP to form composite operators, as shown in Table 2, where $A$ and $B$ are input images of the same size and $c$ is a constant (ranging from $-20$ to 20) stored in the primitive operator. For operators such as ADD, SUB, MUL, etc. that take two images as input, the operations are performed on the pixel-by-pixel basis. In the operators MAX, MIN, MED, MEAN and STDV, $3 \times 3$, $5 \times 5$ or $7 \times 7$ neighborhood are used with equal probability. Operator 16 (MEAN) can be considered as a kind of convolution for low pass filtering and operator 17 (STDV) is a kind of convolution for high pass filtering. Operators 13 (MAX), 14 (MIN) and 15 (MED) can also be considered as convolution operators. We do not include edge operators for several reasons. First, these operators are not primitive and we want to investigate if GP can synthesize effective composite operators or features from simple and domain-independent operations. This is important since without relying on domain knowledge, we can examine the power of a learning algorithm when applied to a variety of images or other areas. Second, edge detection operators can be dissected into the above primitive operators and it is possible for GP to synthesize edge operators or composite operators approximating them if they are very useful to the cur-

Table 2
Seventeen primitive operators

| No. | Operator | Description |
|---|---|---|
| 1 | ADD($A$, $B$) | Add images $A$ and $B$ |
| 2 | SUB($A$, $B$) | Subtract image $B$ from $A$ |
| 3 | MUL($A$, $B$) | Multiply images $A$ and $B$ |
| 4 | DIV($A$, $B$) | Divide image $A$ by image $B$ (if the pixel in $B$ has value 0, the corresponding pixel in the resultant image takes the maximum pixel value in $A$) |
| 5 | MAX2($A$, $B$) | The pixel in the resultant image takes the larger pixel value of images $A$ and $B$ |
| 6 | MIN2($A$, $B$) | The pixel in the resultant image takes the smaller pixel value of images $A$ and $B$ |
| 7 | ADDC($A$) | Increase each pixel value by $c$ |
| 8 | SUBC($A$) | Decrease each pixel value by $c$ |
| 9 | MULC($A$) | Multiply each pixel value by $c$ |
| 10 | DIVC($A$) | Divide each pixel value by $c$ |
| 11 | SQRT($A$) | For each pixel with value $v$, if $v \geq 0$, change its value to $\sqrt{v}$. Otherwise, to $-\sqrt{-v}$ |
| 12 | LOG($A$) | For each pixel with value $v$, if $v \geq 0$, change its value to $\ln(v)$. Otherwise, to $-\ln(-v)$ |
| 13 | MAX($A$) | Replace the pixel value by the maximum pixel value in a $3 \times 3$, $5 \times 5$ or $7 \times 7$ neighborhood |
| 14 | MIN($A$) | Replace the pixel value by the minimum pixel value in a $3 \times 3$, $5 \times 5$ or $7 \times 7$ neighborhood |
| 15 | MED($A$) | Replace the pixel value by the median pixel value in a $3 \times 3$, $5 \times 5$ or $7 \times 7$ neighborhood |
| 16 | MEAN($A$) | Replace the pixel value by the average pixel value of a $3 \times 3$, $5 \times 5$ or $7 \times 7$ neighborhood |
| 17 | STDV($A$) | Replace the pixel value by the standard deviation of pixels in a $3 \times 3$, $5 \times 5$ or $7 \times 7$ neighborhood |

rent object detection task. Finally, the primitive operator library is decoupled from the GP learning system. Edge detection operators can be added in the primitive operator library if they are absolutely needed by the current object detection task.

Some operations used to generate feature images are the same as some primitive operators (see Tables 1 and 2), but there are some differences. Primitive feature images are generated from an original image, so the operations generating primitive feature images are applied to an original image. A primitive operator is applied to a primitive feature image or to the intermediate image output generated by the child node of the node containing this primitive operator. In short, the input image of a primitive operator varies.

### *3.1.3. The fitness measure*

It measures the extent to which the ground-truth and the extracted ROI overlap. The fitness value of a composite operator is computed in the following way. Suppose $G$ and $G'$ are foregrounds in the ground-truth image and the resultant image of the composite operator, respectively. Let $n(X)$ denote the number of pixels within region $X$, then fitness $= n(G \cap G')/n(G \cup G')$. The fitness value is between 0 and 1. If $G$ and $G'$ are completely separated, the value is 0; if $G$ and $G'$ are completely overlapped, the value is 1.

### *3.1.4. Parameters and termination*

The key parameters are the population size $M$, the number of generation $N$, the crossover rate, the mutation rate and the fitness threshold. The GP stops whenever it finishes the pre-specified number of generations or whenever the best composite operator in the population has fitness value greater than the fitness threshold.

## *3.2. Selection, crossover and mutation*

GP searches through the space of composite operator to generate new composite operators, which may be better than the previous ones. By searching through the composite operator space, GP gradually adapts the population of composite operators from generation to generation and improves the overall fitness of the whole population. More importantly, GP may find an exceptionally good composite operator during the search. The search is done by performing selection, crossover and mutation operations. The initial population is randomly generated and the fitness of each individual is evaluated.

### *3.2.1. Selection*

The selection operation involves selecting composite operators from the current population. In this paper, we use tournament selection, where a number of individuals are randomly selected from the current population and the one with the highest fitness value is copied into the new population. The size of tournament is five.

### 3.2.2. *Crossover*

To perform crossover, two composite operators are selected on the basis of their fitness values. The higher the fitness value, the more likely the composite operator is selected for crossover. These two composite operators are called parents. One internal node in each of these two parents is randomly selected, and the two subtrees rooted at these two nodes are exchanged between the parents to generate two new composite operators, called offspring. The offspring are composed of subtrees from their parents. If two composite operators are somewhat effective in detection, then some of their parts probably have some merit. The reason that an offspring may be better than the parents is that recombining randomly chosen parts of somewhat effective composite operators may yield a new composite operator more effective in detection.

It is easy to see that the size of one offspring (i.e., the number of nodes in the binary tree representing the offspring), may be greater than both parents. So if we do not control the size of composite operators by implementing crossover in this simple way, the sizes of composite operators will become larger and larger as GP proceeds. This is the well-known code-bloat problem of GP. It is a very serious problem, since when the size becomes too large, it will take a long time to execute a composite operator, thus greatly reducing the search speed of GP. Further, large-size composite operators may overfit the training data by approximating various noisy components of an image. Although the results on the training image may be very good, the performance on unseen testing images may be bad. Also, large composite operators take up a lot of computer memory. Due to the finite computer resources and the desire to achieve a good running speed (efficiency) of GP, we must limit the size of composite operator by specifying its maximum size. In our previous work [6], if the size of one offspring exceeds the *maximum size* allowed, the crossover operation is performed again until the sizes of both offspring are within the limit. Although this simple method guarantees that the size of composite operator does not exceed the size limit, it is a brutal method since it sets a hard size limit. The hard size limit may greatly restrict the search performed by GP, since after randomly selecting a crossover point in one composite operator, GP cannot select some nodes of the other composite operator as a crossover point in order to guarantee that both offspring do not exceed the size limit. However, restricting the search may greatly reduce the efficiency of GP, making it less likely to find good composite operators.

One may suggest that after two composite operators are selected, GP may perform crossover twice and may each time keep the offspring of smaller size. This method can enforce the size limit and will prevent the sizes of offspring composite operators from growing large. However, GP will now only search the space of these smaller composite operators. With small number of nodes, a composite operator may not capture the characteristics of objects to be detected. How to avoid restricting the GP search while at the same time prevent code-bloat is the key to the success of GP and it is still a subject of intensive research. The key is to find a balance between these two conflicting factors.

In this paper, we set a composite operator size limit to prevent code-bloating, but unlike our previous work, the size limit is a *soft size limit*, so it restricts the GP search less severely than the hard size limit. With soft size limit, GP can select any node in both composite operators as crossover points. If the size of an offspring exceeds the size limit, GP still keeps it and evaluates it later. If the fitness of this large composite operator is the best or very close to the fitness of the best composite operator in the population, it is kept by GP; otherwise, GP randomly selects one of its subtrees of size smaller than the size limit to replace it in the population. In this paper, GP discards any composite operator beyond the size limit unless it is the best one in the population. By keeping the effective composite operators exceeding the size limit, GP enhances the possibility of finding good composite operators, since good composite operators usually contain effective components (subtrees) and these effective components are kept by soft size limit and they may transfer to other composite operators during crossover. Also, by keeping some large composite operators, the size difference between composite operators in the population is widened and this is helpful in reducing the possibility of fitness bloat (in which an increasing number of redundant composite operators in the population evaluate to the same fitness value), although it cannot get rid of it. With hard size limit, many composite operators in the population have size equal or very close to the hard size limit in the later generations of GP. This increases the possibility of fitness bloat. However, large composite operators kept by soft size limit take long time to execute and many of them have redundant branches. By getting rid of the redundant branches, we can reduce the size and running time of composite operators without degrading their performance. But, in order to identify the redundant branches, the fitness of each internal node has to be evaluated and this is a time-consuming process. Moreover, some redundant branches are effective components. They are redundant just because they are in an inhospitable context and their effect is cancelled by other nodes. Eliminating them does no good to the GP search since these effective components may go into other friendly composite operators via crossover operation. Also, composite operators with redundant branches are more resistant to destructive crossover and mutation. Without redundant branches, each part of a composite operator is important to its performance and breaking any component may have a major impact on the performance of the composite operator.

A possible remedy to the problem of redundant branches is that after the GP search terminates, GP or human experts can examine the best composite operator generated by GP carefully, trying to explain the function of each branch and eliminate redundant ones. In this way, GP reduces the size and running time of a composite operator without reducing its performance, making it more efficient in object detection.

#### 3.2.3. *Mutation*

In order to avoid premature convergence, mutation is introduced to randomly change the structure of some individuals to maintain the diversity of the population. Composite operators are randomly selected for mutation. There are three types of mutation invoked with equal probability:

1. Randomly select a node of the binary tree representing the composite operator and replace the subtree rooted at this node, including the node selected, by another randomly generated binary tree.
2. Randomly select a node of the binary tree representing the composite operator and replace the primitive operator stored in the node with another primitive operator of the same parity as the replaced one. The replacing primitive operator is selected at random from all the primitive operators with the same parity as the replaced one.
3. Randomly select two subtrees within the composite operator and swap these two subtrees. Of course, neither of the two subtrees can be the subtree of the other.

### 3.3. *Steady-state and generational genetic programming*

Both steady-state and generational genetic programming are used in this paper. In *steady-state GP*, two parent composite operators are selected on the basis of their fitness for crossover. The children of this crossover replace a pair of composite operators with the smallest fitness values. The two children are executed immediately and their fitness values are recorded. Then another two parent composite operators are selected for crossover. This process is repeated until crossover rate is satisfied. Finally, mutation is applied to the resulting population and the mutated composite operators are executed and evaluated. The above cycle is repeated from generation to generation. In *generational GP*, two composite operators are selected on the basis of their fitness values for crossover and generate two offspring. The two offspring are not put into the current population and would not participate in the following crossover operations on the current population. The above process is repeated until crossover rate is satisfied. Then, mutation is applied to the composite operators in the current population and the offspring from crossover. After mutation is done, selection is applied to the current population to select some composite operators. The number of composite operator selected must meet the condition that after combining with the composite operators from crossover, we get a new population of the same size as the old one. Finally, combine the composite operators from crossover with those selected from the old population to get a new population and the next generation begins. In addition, we adopt an elitism replacement method that keeps the best composite operator from generation to generation.

### 3.3.1. *Steady-state genetic programming*

```
0.  randomly generate population $P$ of size $M$ and
    evaluate each composite operator in $P$.
1.  for gen = 1 to $N$ do//$N$ is the number of generation
2.      keep the best composite operator in $P$.
3.      repeat
4.  select 2 composite operators from $P$ based on their
    fitness values for crossover.
5.  select 2 composite operators with the lowest fitness
    values in $P$ for replacement.
6.  perform crossover operation and let the 2 offspring
    replace the 2 composite operators selected for replacement.
7.  execute the 2 offspring and evaluate their fitness values.
8.  until crossover rate is met.
9.  perform mutation on each composite operator
    with probability of mutation_rate and evaluate
    mutated composite operators.
10. After crossover and mutation, a new population
    $P'$ is generated.
11. let the best composite operator from population $P$
    replace the worst composite operator in $P'$ and let
    $P = P'$.
12. if the fitness value of the best composite operator
    in $P$ is above fitness threshold value then
13.         stop.
        endif.
14. check each composite operator in $P$. if its size exceeds
    the size limit and it is not the best composite
    operator in $P$, replace it with one of its subtrees
    whose size is within the size limit.
        endfor//loop 1.
```

### 3.3.2. *Generational genetic programming*

```
0.  randomly generate populations of size $M$ and evaluate
    each composite operator in $P$.
1.  for gen = 1 to $N$ do//$N$ is the number of generation
2.      keep the best composite operator in $P$.
3.  perform crossover on the composite operators in
    $P$ until crossover rate is satisfied and keep all the
    offspring from crossover separately.
4.  perform mutation on the composite operators in
    $P$ and the offspring from crossover with the probability
    of mutation rate.
5.  perform selection on $P$ to select some composite
    operators. The number of selected composite operator
    must be $M$ minus the number of composite
    operators from crossover.
6.  combine the composite operators from crossover
    with those from $P$ to get a new population $P'$ of
    the same size as $P$.
7.  evaluate offspring from crossover and the mutated
    composite operators.
8.  let the best composite operator from $P$ replace the
    worst composite operator in $P'$ and let $P = P'$.
9.  if the fitness of the best composite operator in $P$
    is above fitness threshold then
10.         stop.
        endif
11. check each composite operator in $P$. If its size exceeds
    the size limit and it is not the best composite
    operator in $P$, replace it with one of its subtrees
    whose size is within the size limit.
        endfor//loop 1.
```

## 4. Experiments

Various experiments are performed to test the efficacy of genetic programming in extracting regions-of-interest from real synthetic aperture radar (SAR) images, infrared images and RGB color images. The size of SAR images is $128 \times 128$, except the tank SAR images whose size is $80 \times 80$, and the size of IR and RGB color images is $160 \times 120$. GP (in Section 4.1, examples 1–5, and Sections 4.2 and 4.5) is not applied to a whole training image, but only to a region or regions carefully selected from a training image, to generate the composite operators. The generated composite operator (with the highest fitness) is then applied to the whole training image and to some other testing images to evaluate it. The advantage of performing training on a small selected region is that it can greatly reduce the training time, making it practical for the GP system to be used as a subsystem of other learning systems, which improve the efficiency of GP by adapting the parameters of GP system based on its performance. Our experiments show that if the training regions are carefully selected from the training images, the best composite operator generated by GP is effective. In the following experiments in Sections 4.1, 4.2 and 4.5, the parameters are: population size (100), the number of generations (70), the fitness threshold value (1.0), the crossover rate (0.6), the mutation rate (0.05), the soft size limit of composite operators (30), and the segmentation threshold (0). In each experiment, GP is invoked 10 times with the same parameters and the same training region(s). The coordinate of the upper left corner of an image is (0, 0). The ground-truth is used only during training, it is not needed during testing. We use it in testing only for evaluating the performance of the composite operator on testing images. The size, orientation or shape of the objects in testing images is different from those in the training images.

## 4.1. SAR images

Five experiments are performed with real SAR images. The experimental results from 1 run and the average performance of 10 runs are reported in Table 3. We select the run in which GP finds the best composite operator among the composite operators found in all 10 runs. The first two rows show the fitness value of the best composite operator and the population fitness value (average fitness value of all the composite operators in the population) on training region(s) in the initial and final generations in the selected run. The numbers in the parenthesis in the "$f_{op}$" columns are the fitness values of the best composite operators on the whole training image (numbers with an asterisk in superscript) and other testing images in their entirety. The last two rows show the average values of the above fitness values over all 10 runs. The regions extracted during the training and testing by the best composite operator from the selected run are shown in the following examples.

### 4.1.1. Example 1—road extraction

Three images contain road, the first one contains horizontal paved road and field (Fig. 1(a)); the second one contains unpaved road and field (Fig. 8(a)); the third one contains vertical paved road and grass (Fig 8(d)). Training is done on the training regions of training image shown in Fig. 1(a) and testing is performed on the whole training image and testing images. There are two training regions, locating from (5, 19) to (50, 119) and from (82, 48) to (126, 124), respectively. Fig. 1(b) shows the ground-truth provided by the user and the training regions. The white region corresponds to the road and only the training regions of the ground-truth are used in the evaluation during the training. Fig. 2 shows the 16 primitive feature images of the training image.

The generational GP is used to synthesize a composite operator to extract the road and the results of the sixth run are reported. The fitness value of the best composite operator in the initial population is 0.68 and the population fitness value is 0.28. The fitness value of the best composite operator in the final population is 0.95 and the population fitness value is 0.67. Fig. 1(c) shows the output image of the best composite operator on the whole training image and Fig. 1(d) shows the binary image after segmentation. The output image has both positive pixels in brighter shade and negative pixels in darker shade. Positive pixels belong to the region to be extracted. The fitness value of the extracted ROI is 0.93. The best composite operator has 17 nodes and its depth is 16. It has only one leaf node containing 5 × 5 median image. The median image is less noisy, since median filtering is effective in eliminating speckle noises. The best composite operator is shown in Fig. 3, where PFIM14 is 5×5 median image. Fig. 4 shows how the average fitness of the best composite operator and average fitness of population over all 10 runs change as GP explores the composite operator space. Unlike [6], where the population fitness

Table 3
The performance of our approach on various examples of SAR images

| | Example 1—road | | Example 2—lake | | Example 3—river | | Example 4—field | | Example 5—tank | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $f_{op}$ | $f_p$ | $f_{op}$ | $f_p$ | $f_{op}$ | $f_p$ | $f_{op}$ | $f_p$ | $f_{op}$ | $f_p$ |
| $f_{initial}$ | 0.68 | 0.28 | 0.56 | 0.32 | 0.65 | 0.18 | 0.53 | 0.39 | 0.51 | 0.16 |
| $f_{final}$ | 0.95 (0.93*, 0.9, 0.93) | 0.67 | 0.97 (0.93*, 0.98) | 0.93 | 0.90 (0.71*, 0.83) | 0.85 | 0.78 (0.89*, 0.80) | 0.64 | 0.88 (0.88*, 0.84) | 0.80 |
| Average $f_{initial}$ | 0.55 | 0.27 | 0.59 | 0.32 | 0.48 | 0.18 | 0.54 | 0.37 | 0.61 | 0.17 |
| Average $f_{final}$ | 0.83 | 0.60 | 0.95 | 0.92 | 0.85 | 0.77 | 0.76 | 0.59 | 0.86 | 0.68 |

$f_{op}$: fitness of the best composite operator; $f_p$: fitness of population; $f_{initial}$: fitness in the initial generation; $f_{final}$: fitness in the final population; an asterisk in superscript indicates fitness on training images.

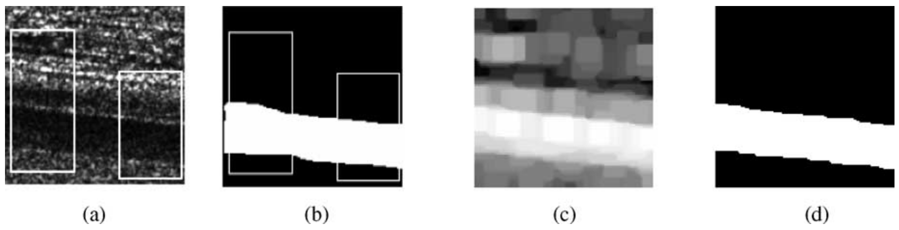

Fig. 1. Training SAR image containing road: (a) paved road vs. field; (b) ground-truth; (c) composite feature image; (d) ROI extracted.

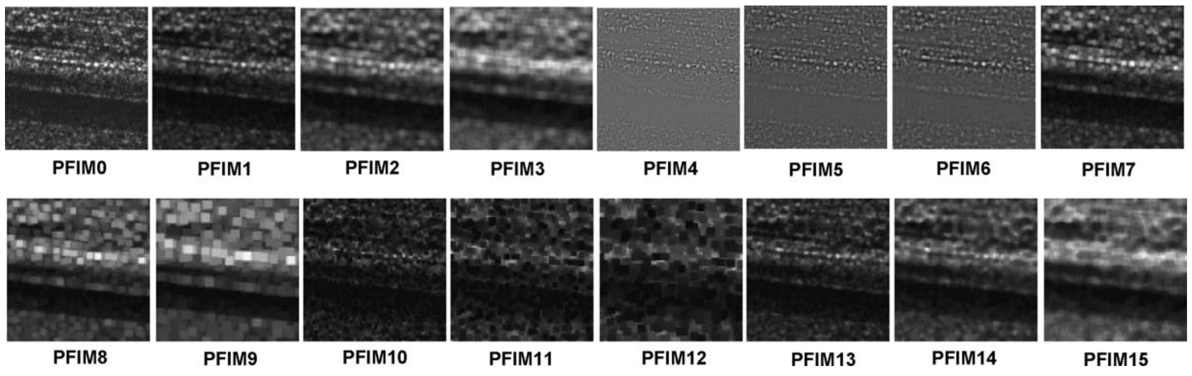

Fig. 2. Sixteen primitive feature images of training SAR image containing road.

approaches the fitness of the best composite operator as GP proceeds, in Fig. 4, population fitness is much lower than that of best composite operator even at the end of GP search. It is reasonable, since we do not restrict the selection of crossover points. The population fitness is not important since only the best composite operator is used in testing. If GP finds one effective composite operator, the GP learning is successful. The large difference between the fitness of the best composite operator and the population indicates that the diversity of the population is always maintained during GP search, which is very helpful in preventing premature convergence.

Ten best composite operators are learned in 10 runs. After computing the percentage of each primitive operator and primitive feature image among the total number of internal nodes (representing primitive operators) and the total number of leaf nodes (representing primitive feature images) of these ten best composite operators, we get the utility of these primitive operators and primitive feature images, which is shown in Fig. 5(a) and (b). MED (primitive operator 15) and PFIM5 (the sixth primitive feature image) have the highest frequency of utility. Fig. 6 shows the output image of each node of the best composite operator shown in Fig. 3. From left to right and top to bot-

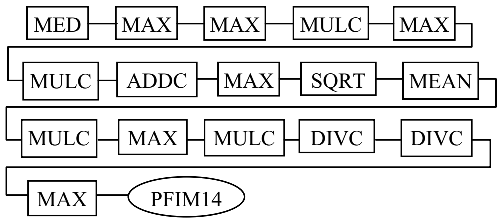

Fig. 3. Learned composite operator tree.

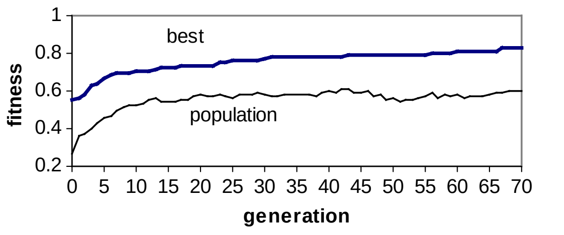

Fig. 4. Fitness vs. generation (road vs. field).

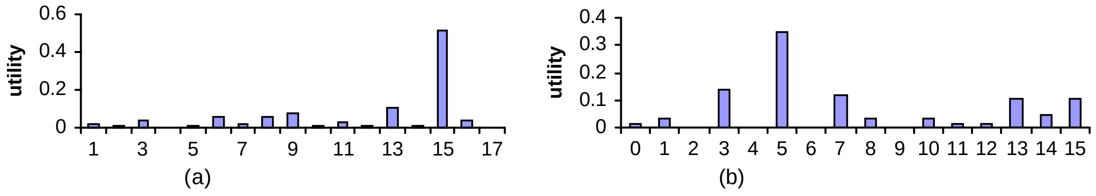

Fig. 5. Utility of primitive operators and primitive feature images: (a) primitive operator; (b) primitive feature image.

tom, the images correspond to nodes sorted in the pre-traversal order of the binary tree representing the best composite operator. The output of the root node is shown in Fig. 1(c), so Fig. 6 shows the outputs of other nodes. The primitive operators in Fig. 6 are connected by arrow. The operator at the tail of an arrow provides input to the operator at the head of the arrow. After segmenting the output image of a node, we get the ROI (shown as the white region) extracted by the corresponding subtree rooted at the node. The extracted ROIs and their fitness values are shown in Fig. 7. If an output image of a node has no positive pixel (for example, the output of MEAN primitive operator), nothing is extracted and the fitness value is 0; if an output image has positive pixels only (for example, PFIM14 has positive pixels only), everything is extracted and the fitness is 0.25. The output of the root node storing primitive operator MED is shown in Fig. 1(d).

We applied the composite operator obtained in the above training to the other two real SAR images shown in Fig. 8(a) and (d). Fig. 8(b) and (e) show the output of the composite operator and Fig. 8(c) shows the region extracted from Fig. 8(a). The fitness value of the region is 0.90. Fig. 8(f) shows the region extracted from Fig. 8(d). The fitness value of the region is 0.93.

### 4.1.2. Example 2—lake extraction

Two SAR images contain lake (Figs. 9(a) and 10(a)), the first one contains a lake and field, and the second one contains a lake and grass. Fig. 9(a) shows the original training image containing lake and field and the training region from (85, 85) to (127, 127). Fig. 5(b) shows the ground-truth provided by the user. The white region corresponds to the lake to be extracted. Fig. 10(a) shows the image containing lake and grass.

The steady-state GP is used to generate the composite operator and the results of the ninth run are reported. The fitness value of the best composite operator in the initial population is 0.56 and the population fitness value is 0.32. The fitness value of the best composite operator in the final population is 0.97 and the population fitness value is 0.93. Fig. 9(c) shows the output image of the best composite operator on the

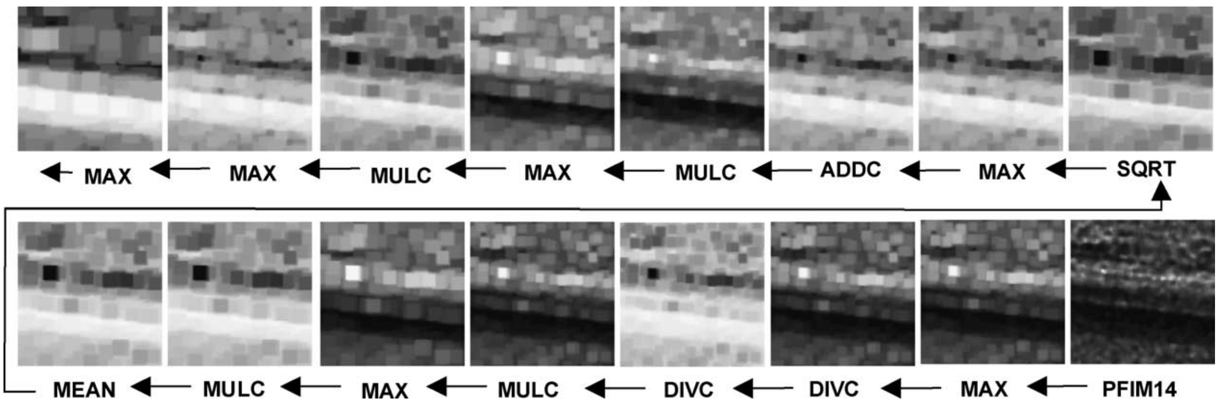

Fig. 6. Feature images output by the nodes of the best composite operator.

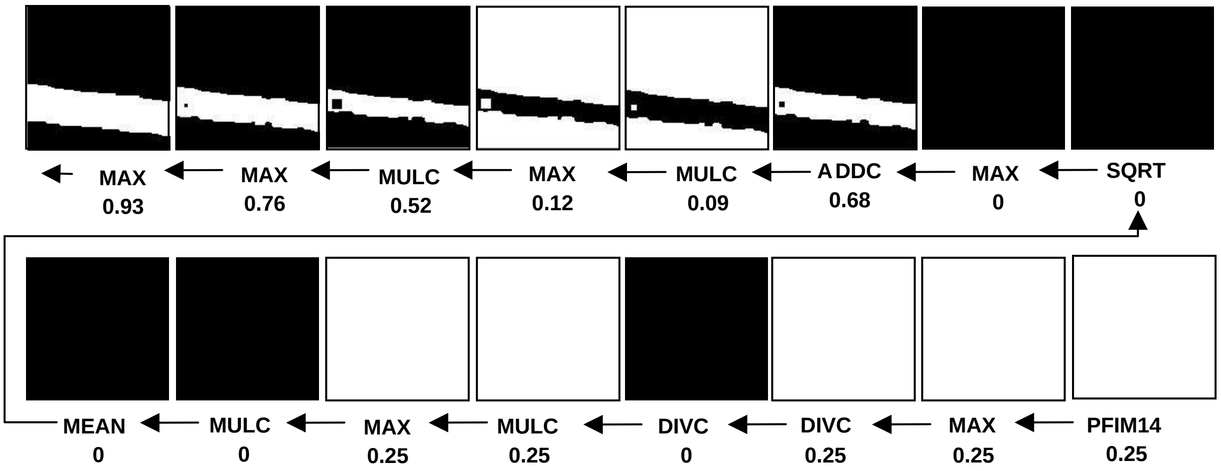

Fig. 7. ROI extracted from the output images of nodes of the best composite operator (the fitness value is shown for the entire image).

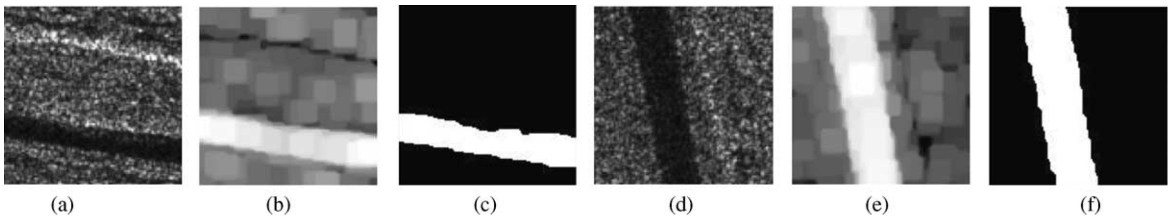

Fig. 8. Testing SAR images containing road: (a) unpaved road vs. field; (b and e) composite feature image; (c and f) ROI extracted; (d) paved road vs. grass.

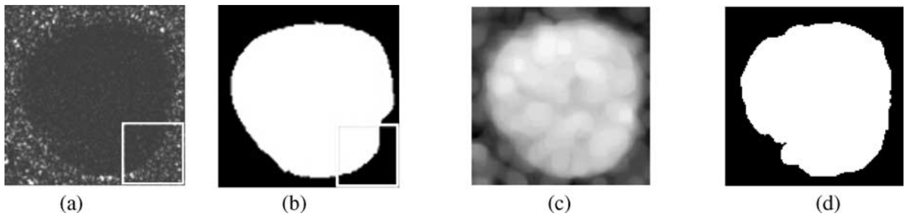

Fig. 9. Training SAR image containing lake: (a) lake vs. field; (b) ground-truth; (c) composite feature image; (d) ROI extracted.

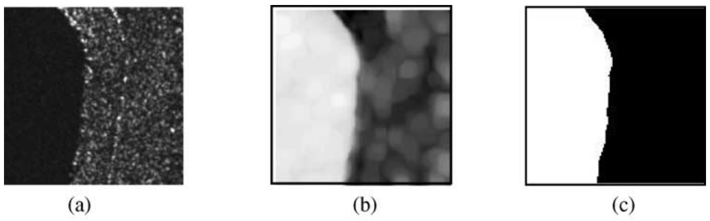

Fig. 10. Testing SAR image containing lake: (a) lake vs. grass; (b) composite feature image; (c) ROI extracted.

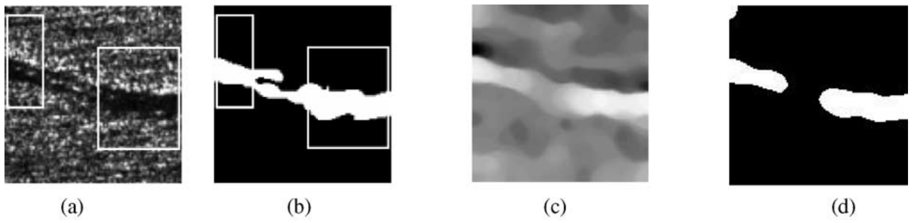
Fig. 11. Training SAR image containing river: (a) river vs. field; (b) ground-truth; (c) composite feature image; (d) ROI extracted.

whole training image and Fig. 9(d) shows the binary image after segmentation. The fitness value of the extracted ROI is 0.93.

We apply the composite operator to the testing image containing lake and grass. Fig. 10(b) shows the output of the composite operator and Fig. 10(c) shows the region extracted from Fig. 10(a). The fitness of the region is 0.98.

### 4.1.3. *Example 3—river extraction*

Two SAR images contain river and field. Fig. 11(a) and (b) show the original training image and the ground-truth provided by the user. The white region in Fig. 11(b) corresponds to the river to be extracted. The training regions are from (68, 31) to (126, 103) and from (2, 8) to (28, 74). The testing SAR image is shown in Fig. 14(a).

The steady-state GP was used to generate the composite operator and the results from the fourth run are reported. The fitness value of the best composite operator in the initial population is 0.65 and the population fitness value is 0.18. The fitness value of the best composite operator in the final population is 0.90 and the population fitness value is 0.85. Fig. 11(c) shows the output image of the best composite operator on the whole training image and Fig. 11(d) shows the binary image after segmentation. The fitness value of the extracted ROI is 0.71. The best composite operator has 29 nodes and its depth is 19. It has five leaf nodes and all contain $7 \times 7$ median image. There are more than 10 MED operators that are very useful in eliminating speckle noises. It is shown in Fig. 12. Fig. 13 shows how the average fitness of the best composite operator and average fitness of population over all 10 runs change as GP explores the composite operator space.

We apply the composite operator to the testing image containing a river and field. Fig. 14(b) shows the output of the composite operator and Fig. 14(c) shows the region extracted from Fig. 14(a) and the fitness value of the region is 0.83. There are some islands in the river and these islands along with part of the river around them are not extracted.

### 4.1.4. *Example 4—field extraction*

Two SAR images contain field and grass. Fig. 15(a) and (b) show the original training image and the ground-truth. The training regions are from (17, 3) to (75, 61) and from (79, 62) to (124, 122). Extracting field from a SAR image containing field and grass is the most difficult task among the five experiments, since the grass and field are similar to each other and some small regions between grassy areas are actually field pixels.

The generational GP was used to generate the composite operator and the results from the second run are

```
(MED (MED (MED (ADD (MED (STDV (MED
PFIM15))) (MED (MED (MED (MED (MED
(MIN2 (MED PFIM15) (MED (MED (MED
(MIN2 (MED PFIM15) (MED (MED (MIN2
PFIM15 (SUBC (DIVC PFIM15)))))))))))))))))))
```

Fig. 12. Learned composite operator tree in LISP notation.

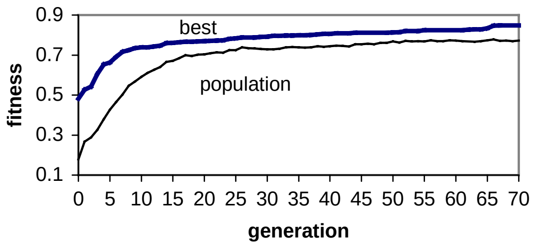

Fig. 13. Fitness vs. generation (river vs. field).

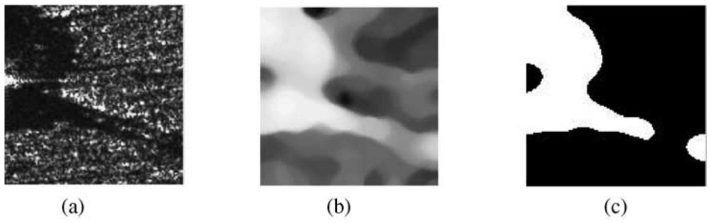
Fig. 14. Testing SAR image containing river: (a) river vs. field; (b) composite feature image; (c) ROI extracted.

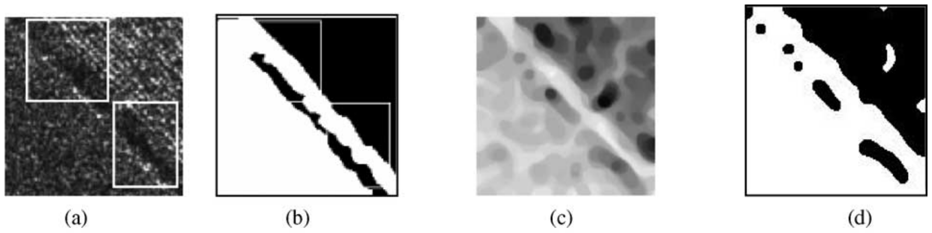
Fig. 15. Training SAR image containing field: (a) field vs. grass; (b) ground-truth; (c) composite feature image; (d) ROI extracted.

reported. The fitness value of the best composite operator in the initial population is 0.53 and the population fitness value is 0.39. The fitness value of the best composite operator in the final population is 0.78 and the population fitness value is 0.64. Fig. 15(c) shows the output image of the best composite operator on the whole training image and Fig. 15(d) shows the binary image after segmentation. The fitness value of the extracted ROI is 0.89.

We apply the composite operator to the testing image containing field and grass shown in Fig. 16(a). Fig. 16(b) shows the output of the composite operator and Fig. 16(c) shows the region extracted from Fig. 16(a). The fitness value of the region is 0.80.

### 4.1.5. *Example 5—tank extraction*

We use SAR images of T72 tank that are taken under different depression and azimuth angles and the size of the images is $80 \times 80$. The training image contains T72 tank under depression angle 17° and azimuth angle 135°, which is shown in Fig. 17(a). The training region is from (19, 17) to (68, 66). The testing SAR image contains a T72 tank under depression angle 20° and azimuth angle 225°, which is shown in Fig. 20(a). The ground-truth is shown in Fig. 17(b).

The generational GP is applied to synthesize composite operators for tank detection and the results from the first run are reported. The fitness value of the best composite operator in the initial population is 0.51 and the population fitness value is 0.16. The fitness value of the best composite operator in the final population is 0.88 and the population fitness value is 0.80. Fig. 17(c) shows the output image of the best composite operator on the whole training image and Fig. 17(d) shows the binary image after segmentation. The fitness value of the extracted ROI is 0.88. The best composite op-

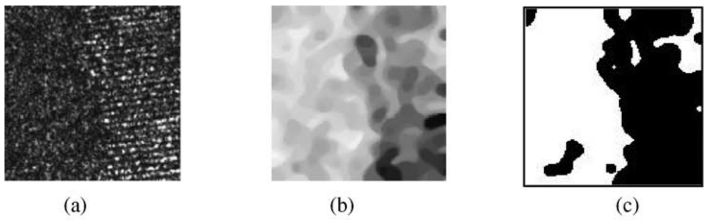
Fig. 16. Testing SAR image containing field: (a) field vs. grass; (b) composite feature image; (c) ROI extracted.

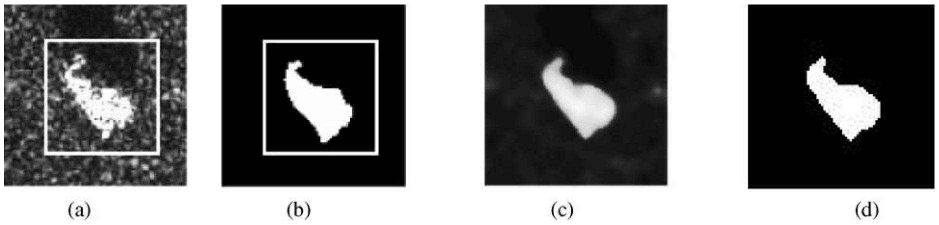

Fig. 17. Training SAR image containing field: (a) T72 tank; (b) ground-truth; (c) composite feature image; (d) ROI extracted.

```
(MED (SQRT (MULC (MULC (SUBC (MULC
(SQRT (SUBC (SQRT PFIM2))))))))
```

Fig. 18. Learned composite operator tree in LISP notation.

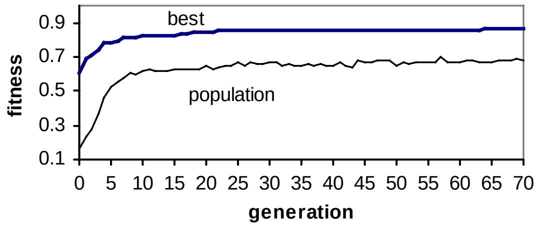

Fig. 19. Fitness vs. generation (T72 tank).

erator has 10 nodes and its depth is 9. It has only one leaf node, which contains the 5 × 5 mean image. It is shown in Fig. 18. Fig. 19 shows how the average fitness of the best composite operator and average fitness of population over all 10 runs change as GP proceeds.

We apply the composite operator to the testing image containing T72 tank under depression angle 20° and azimuth angle 225°. Fig. 20(b) shows the output of the composite operator and Fig. 20(c) shows the region corresponding to the tank. The fitness of the extracted ROI is 0.84. Our results show that GP is very much capable of synthesizing composite operators for target detection. With more and more SAR images collected by satellites and airplanes, it is impractical for human experts to scan each SAR image to find targets. Applying the synthesized composite operators on these images, regions containing potential targets can be quickly detected and passed on to automatic target recognition systems or to human experts for further examination. Concentrating on the regions-of-interest, the human experts and recognition systems can perform recognition task more effectively and more efficiently.

Note that composite operators shown in Figs. 3 and 18 may be called as "processing chains," which is a kind of binary tree in which each internal node has only one child. Most of the composite operators learned by GP in our experiments are not processing chains. Processing chains are simpler than ordinary binary trees. It is good for GP to learn simple processing chains to produce good ROI extraction results.

## 4.2. IR and RGB color images

One experiment is performed with infrared images and two are performed with RGB color images. The

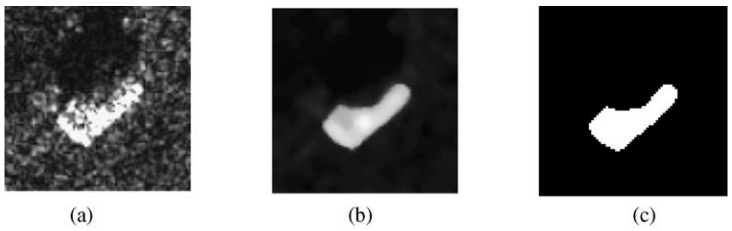

Fig. 20. Testing SAR image containing tank: (a) T72 tank; (b) composite feature image; (c) ROI extracted.

Table 4
The performance of our approach on examples of IR and RGB color images

| | IR image—people | | Color image—car | | Color image—SUV | |
|---|---|---|---|---|---|---|
| | $f_{\text{op}}$ | $f_{\text{p}}$ | $f_{\text{op}}$ | $f_{\text{p}}$ | $f_{\text{op}}$ | $f_{\text{p}}$ |
| $f_{\text{initial}}$ | 0.56 | 0.23 | 0.35 | 0.18 | 0.33 | 0.22 |
| $f_{\text{final}}$ | 0.93 (0.85*, 0.84, 0.81, 0.86) | 0.79 | 0.84 (0.82*, 0.76) | 0.79 | 0.69 (0.69*, 0.58) | 0.65 |
| Average $f_{\text{initial}}$ | 0.59 | 0.21 | 0.47 | 0.18 | 0.34 | 0.21 |
| Average $f_{\text{final}}$ | 0.85 | 0.65 | 0.72 | 0.67 | 0.61 | 0.56 |

$f_{\text{op}}$: fitness of the best composite operator; $f_{\text{p}}$: fitness of population; $f_{\text{initial}}$: fitness in the initial generation; $f_{\text{final}}$: fitness in the final population; an asterisk in superscript indicates fitness on training images.

experimental results from 1 run and the average performance of 10 runs are reported in Table 4. As we did in Section 4.1, we select the run in which GP finds the best composite operator among the composite operators found in all the 10 runs. The regions extracted during the training and testing by the best composite operator from the selected run are shown in the following examples.

### 4.2.1. People extraction in IR images

In IR images, pixel values correspond to the temperature in the scene. We have four IR images with one used in training and the other three used in testing. Fig. 21(a) and (b) show the training image and the ground-truth. Two training regions are from (59, 9) to (106, 88) and from (2, 3) to (21, 82), respectively. The left training region contains no pixel belonging to the person. The reason for selecting it during the training is that there are major pixel intensity changes among the pixels in this region. Nothing in this region should be detected. The fitness of composite operator on this region is defined as one minus the percentage of pixels detected in the region. If nothing is detected, the fitness value is 1.0. Averaging the fitness values of the two training regions, we get the fitness during training. When the learned composite operator is applied to the whole training image, the fitness is computed as a measurement of the overlap between the ground-truth and the extracted ROI, as we did in the previous experiments. Three testing IR images are shown in Fig. 24(a), (d) and (g).

The generational GP is applied to synthesize composite operators for person detection and the results from the third run are reported. The fitness value of the best composite operator in the initial population is 0.56 and the population fitness value is 0.23. The fitness value of the best composite operator in the final population is 0.93 and the population fitness value is 0.79. Fig. 21(c) shows the output image of the best composite operator on the whole training image and Fig. 21(d) shows the binary image after segmentation. The fitness value of the extracted ROI is 0.85. The best composite operator has 28 nodes and its depth is 13. It has nine leaf nodes and is shown in Fig. 22. Fig. 23 shows how the average fitness of the best composite operator and average fitness of population over all the 10 runs change as GP proceeds.

We apply the composite operator to the testing images shown in Fig. 24. Fig. 24(b), (e) and (h) show the output of the composite operator and Fig. 24(c), (f) and (i) show the ROI extracted. Their fitness values are 0.84, 0.81 and 0.86, respectively.

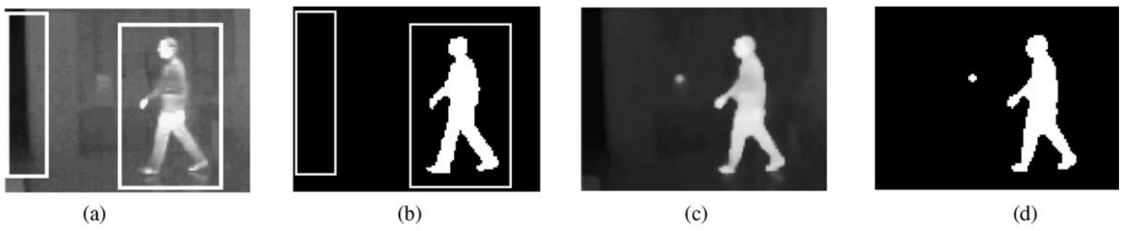

Fig. 21. Training IR image containing person: (a) person; (b) ground-truth; (c) composite feature image; (d) ROI extracted.

```
(SQRT (SQRT (SUBC (SQRT (MAX2 (MAX2
PFIM1 (SUB (MAX2 PFIM14 PFIM15) (DIV
(MULC (SQRT (MAX (MAX (ADD PFIM12
PFIM15))))) PFIM9))) (DIV (MULC (SQRT
(MAX (ADD PFIM12 PFIM9)))) PFIM9))))))
```

Fig. 22. Learned composite operator tree in LISP notation.

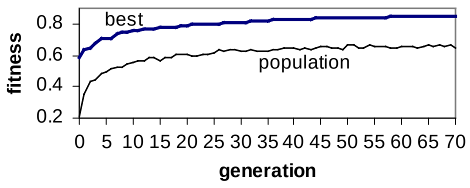

Fig. 23. Fitness vs. generation (person).

#### 4.2.2. *Car extraction in RGB color images*

GP is applied to learn features to detect car in RGB color images. Unlike previous experiments, the primitive feature images in this experiment are red, green and blue planes of RGB color image. Fig. 25(a)–(c) show the red, green and blue planes of the training image. The ground-truth is shown in Fig. 25(d). The training region is from (21, 3) to (91, 46).

The steady-state GP is applied to synthesize composite operators for car detection and the results from the fourth run are reported. The fitness value of the best composite operator in the initial population is 0.35 and the population fitness value is 0.18. The fitness value of the best composite operator in the final population is 0.84 and the population fitness value is 0.79. Fig. 25(e) shows the output image of the best composite operator on the whole training image and Fig. 25(f) shows the binary image after segmentation. The fitness value of the extracted ROI is 0.82. The best composite operator has 44 nodes and its depth is 21. It has 10 leaf nodes with one containing green plane and the others containing blue plane. It is shown in Fig. 26, where PFG means green plane and PFB means blue plane. Note that only green and blue planes are used by the composite operator. Fig. 27 shows how the average fitness of the best composite operator and

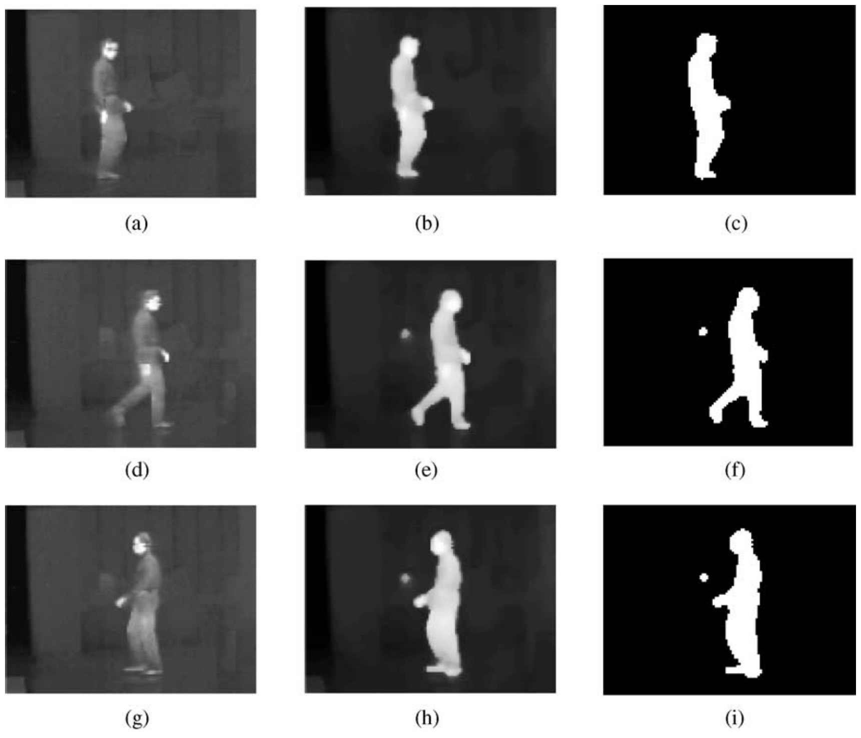

Fig. 24. Testing IR images containing person: (a, d and g) person; (b, e and h) composite feature image; (c, f and i) ROI extracted.

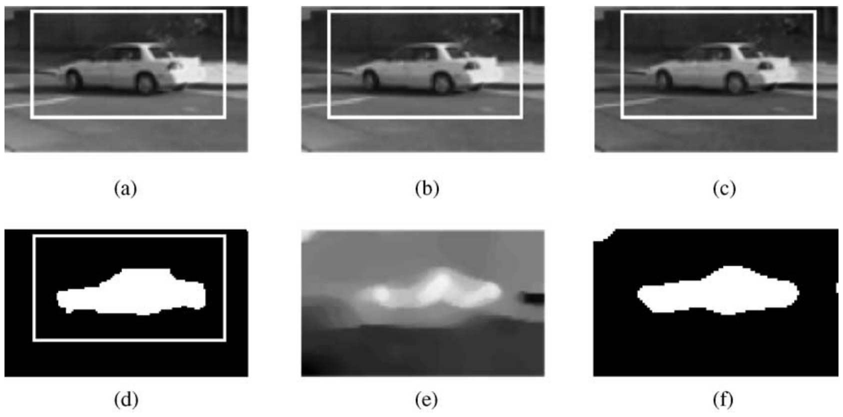

Fig. 25. Training RGB color image containing car: (a) red plane; (b) green plane; (c) blue plane; (d) ground-truth; (e) composite feature image; (f) ROI extracted.

```
(MED (MED (MED (MULC (MUL (SUB (MIN
(MEAN (MAX2 (MED (ADDC (MAX2 (ADDC
(ADDC (MED (MAX2 (MED (MED (MAX2 (MED
(ADDC PFB)) PFB))) PFB)))) PFB))) (MED
PFG)))) (ADDC (MAX2 (ADDC (ADDC (MED
(MAX2 (MED (MED (MAX2 (MED (ADDC PFB))
PFB))) PFB)))) PFB))) (ADDC PFB))))))
```

Fig. 26. Learned composite operator tree in LISP notation.

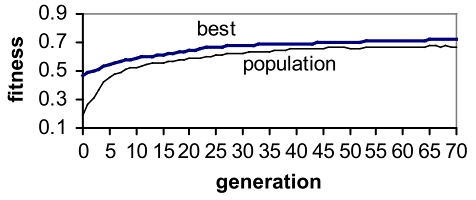

Fig. 27. Fitness vs. generation (car).

average fitness of population over all 10 runs change as GP runs.

We apply the composite operator to the testing image whose red plane is shown in Fig. 28(a). Fig. 28(b) shows the output of the composite operator and Fig. 28(c) shows the ROI extracted. The fitness value of extracted ROI is 0.76.

### 4.2.3. SUV extraction in RGB color images

In this section, GP is applied to learn features to detect SUV (sports utility vehicle) in RGB color images. The images containing a SUV have more complicated background than the images containing the car, increasing the difficulty in SUV detection. This will be a difficult example for any segmentation technique in computer vision and pattern recognition. Fig. 29(a)–(c) show the red, green and blue planes of the training image and Fig. 29(d) shows the ground-truth. The training region is from (20, 21) to (139, 100). Fig. 29(f) and (g) show the red plane and the ground-truth of the testing image.

The steady-state GP is applied to synthesize composite operators for SUV detection and the results from the fourth run are reported. The fitness value of the best composite operator in the initial population is 0.33 and the population fitness value is 0.22. The

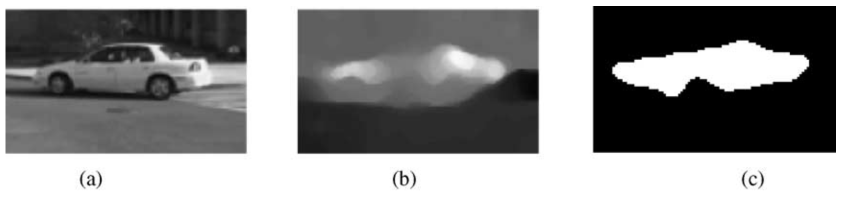

Fig. 28. Testing RGB color image containing car: (a) red plane; (b) composite feature image; (c) ROI extracted.

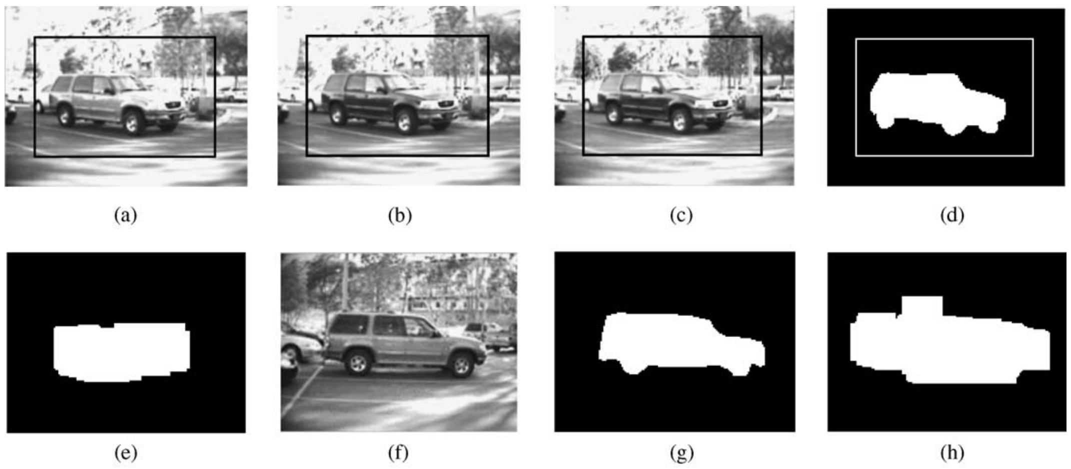

Fig. 29. Training and testing RGB color image containing SUV: (a and f) red plane; (b) green plane; (c) blue plane; (d and g) ground-truth; (e and h) ROI extracted.

fitness value of the best composite operator in the final population is 0.69 and the population fitness value is 0.65. Fig. 29(e) and (h) show the ROI extracted by the best composite operator from training and testing images. The fitness values of the extracted ROIs are 0.69 and 0.58, respectively. The extracted ROIs are not very satisfactory, since the shapes of ROIs differ from the shapes of vehicles in images. However, the extracted ROIs contain SUVs in the training and testing images, which means the locations of the vehicle are correctly detected.

### 4.3. Comparison with previous results

In [6], we applied genetic programming to learn composite operators for object detection. This paper is an advancement to our previous work. The major differences between the method presented here and that in [6] are:

(1) Unlike [6], where a whole training image is used during training (image-based GP), GP runs on carefully selected region(s) (region-based GP) in this paper to reduce the training time.
(2) Hard size limit on the composite operator is replaced by soft size limit in this paper. This removes the restriction on the selection of crossover point in the parent composite operators to improve the search efficiency of GP, as stated in Section 3.2.
(3) Only the first mutation type in Section 3.2 and only the first seven primitive feature images are used in [6]. With more mutation types and more primitive feature images used, the diversity of the composite operator population can be further increased.

We summarize the experimental results on SAR images in [6] for the purpose of comparison. The parameters are: population size (100), the number of generations (100), the fitness threshold value (1.0), the crossover rate (0.6), the mutation rate (0.1), the maximum size (number of internal nodes) of composite operator (30), and the segmentation threshold (0). In each experiment, GP is invoked 10 times with the same parameters. The experimental results from 1 run and the average performance of 10 runs are reported in Table 5. We select the run in which GP finds the best composite operator among the composite operators found in all 10 runs to report. In this table, the numbers in the parenthesis in the "$f_{op}$" columns are the fitness values of the best composite operators on the testing SAR images.

#### 4.3.1. Road extraction

Fig. 1(a) shows the training image and Fig. 8(a) and (d) show the testing images. The generational GP was used to generate a composite operator to extract the road. The fitness value of the best composite operator in the initial population is 0.47 and the population

Table 5
The performance of genetic programming on various examples of SAR images

| | Road | | Lake | | River | | Field | |
|---|---|---|---|---|---|---|---|---|
| | $f_{\mathrm{op}}$ | $f_{\mathrm{p}}$ | $f_{\mathrm{op}}$ | $f_{\mathrm{p}}$ | $f_{\mathrm{op}}$ | $f_{\mathrm{p}}$ | $f_{\mathrm{op}}$ | $f_{\mathrm{p}}$ |
| $f_{\mathrm{initial}}$ | 0.47 | 0.19 | 0.65 | 0.42 | 0.43 | 0.21 | 0.62 | 0.44 |
| $f_{\mathrm{final}}$ | 0.92* (0.92, 0.89) | 0.89 | 0.93* (0.92) | 0.92 | 0.74* (0.84) | 0.68 | 0.87* (0.68) | 0.86 |
| Average $f_{\mathrm{initial}}$ | 0.47 | 0.18 | 0.73 | 0.39 | 0.37 | 0.11 | 0.65 | 0.41 |
| Average $f_{\mathrm{final}}$ | 0.81 | 0.76 | 0.92 | 0.87 | 0.68 | 0.58 | 0.84 | 0.77 |

$f_{\mathrm{op}}$: fitness of the best composite operator; $f_{\mathrm{p}}$: fitness of population; $f_{\mathrm{initial}}$: fitness in the initial generation; $f_{\mathrm{final}}$: fitness in the final population; an asterisk in superscript indicates fitness on training images.

fitness value is 0.19. The fitness value of the best composite operator in the final population is 0.92 and the population fitness value is 0.89. Fig. 30(a) shows the output image of the best composite operator in the final population and Fig. 30(b) shows the extracted ROI. We applied the composite operator obtained in the above training to the two testing SAR images. Fig. 30(c) and (d) show the output image of the composite operator and the ROI extracted from Fig. 8(a). The fitness value of the extracted ROI is 0.92. Fig. 30(e) and (f) show the output image of the composite operator and the ROI extracted from Fig. 8(d). The fitness value of the extracted ROI is 0.89.

### 4.3.2. *Lake extraction*

Fig. 9(a) shows the training image and Fig. 10(a) shows the testing image. The steady-state GP was used to generate the composite operator. The fitness value of the best composite operator in the initial population is 0.65 and the population fitness value is 0.42. The fitness value of the best composite operator in the final population is 0.93 and the population fitness value is 0.92. Fig. 31(a) shows the output image of the best composite operator in the final population and Fig. 31(b) shows the extracted ROI. We applied the composite operator to the testing SAR image. Fig. 31(c) and (d) show the output image of the composite operator and the extracted ROI with fitness value 0.92. In Fig. 31(a) and (c), pixels in the small dark regions have very low pixel values (negative values with very large absolute values), thus making many pixels appear bright, although some of them have negative pixel values.

### 4.3.3. *River extraction*

Fig. 11(a) shows the training image and Fig. 14(a) shows the testing image. The steady-state GP was used to generate the composite operator. The fitness value of the best composite operator in the initial population is 0.43 and the population fitness value is 0.21. The fitness value of the best composite operator in the final population is 0.74 and the population fitness value is 0.68. Fig. 32(a) shows the output image of the best composite operator in the final population and Fig. 32(b) shows the extracted ROI. We applied the composite operator to the testing image. Fig. 32(c) and (d) show the output image of the composite operator and the extracted ROI with fitness value 0.84. Like Fig. 31(c), pixels in the small dark region have very low pixel values (negative values with very large ab-

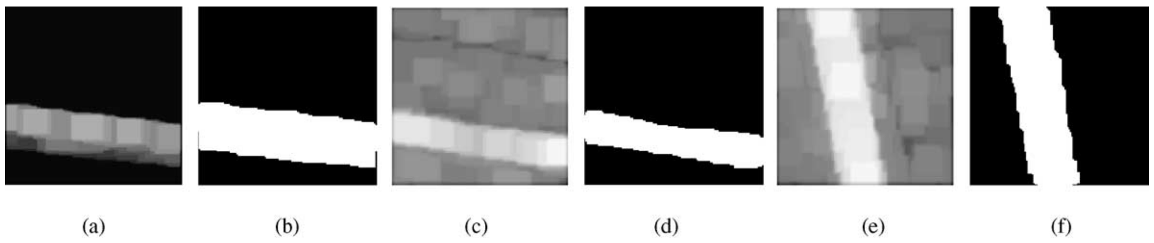

Fig. 30. Results on SAR images containing road: (a, c and e) composite feature image; (b) ROI extracted from Fig. 1(a); (d) ROI extracted from Fig. 8(a); (f) ROI extracted from Fig. 8(d).

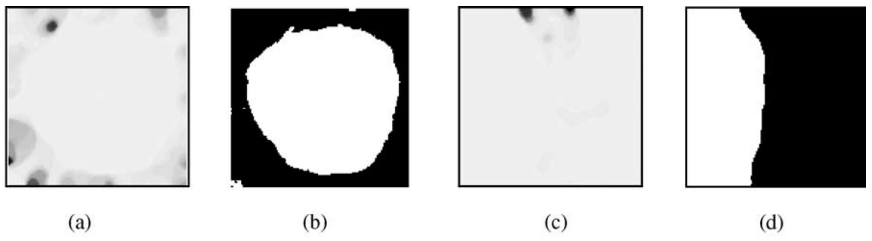
Fig. 31. Results on SAR images containing lake: (a and c) composite feature image; (b) ROI extracted from Fig. 9(a); (d) ROI extracted from Fig. 10(a).

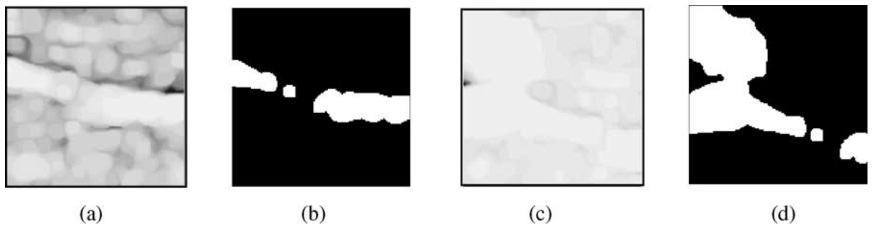
Fig. 32. Results on SAR images containing river: (a and c) composite feature image; (b) ROI extracted from Fig. 11(a); (d) ROI extracted from Fig. 14(a).

solute values), thus making many pixels with negative pixel values appear bright,

### 4.3.4. Field extraction

Fig. 15(a) shows the training image and Fig. 16(a) shows the testing image. The generational GP was used to generate the composite operator. The fitness value of the best composite operator in the initial population is 0.62 and the population fitness value is 0.44. The fitness value of the best composite operator in the final population is 0.87 and the population fitness value is 0.86. Fig. 33(a) shows the output image of the best composite operator in the final population and Fig. 33(b) shows the extracted ROI. We applied the composite operator to the testing image. Fig. 33(c) and (d) show the output image of the composite operator and the extracted ROI with fitness value 0.68.

From Tables 3 and 5 and associated figures, it can be seen that if the carefully selected training regions represent the characteristics of training images, the composite operators learned by GP running on training regions are effective in extracting the ROIs containing the object and their performances are comparable to the performances of composite operators learned by

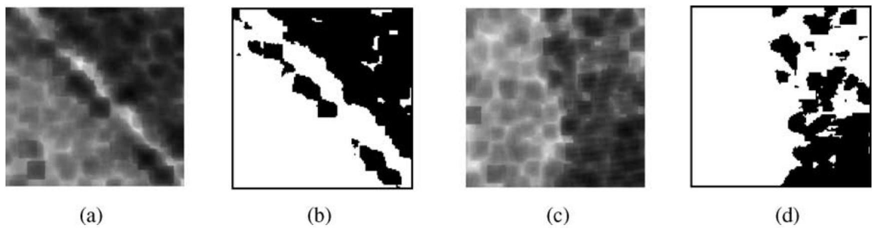
Fig. 33. Results on SAR images containing field: (a and c) composite feature image; (b) ROI extracted from Fig. 15(a); (d) ROI extracted from Fig. 16(a).

Table 6
Average running time of region GP and image GP

| | Road | Lake | River | Field |
|---|---|---|---|---|
| Region-based GP | 12876 | 2263 | 6560 | 9685 |
| Image-based GP | 23608 | 9120 | 66476 | 21485 |

GP running on whole training images. By running on the selected regions, the training time is greatly reduced. Table 6 shows the average running time of GP running on selected regions (region GP) and GP running on the whole training images (image GP) over all 10 runs and the time is measured in second. Since the number of generation in [6] is 100 and the number of generation in this paper is 70, the running time of "image GP" in Table 6 is actually the running time of "image GP" times 0.7. It can be seen that the training time using selected training regions is much shorter than that using the whole image.

### 4.4. Comparison with a traditional ROI extraction algorithm

To show the effectiveness of composite operators in ROI extraction, they are compared with a traditional ROI extraction algorithm. The traditional ROI extraction algorithm uses a threshold value to segment the image into foreground and background. The region consisting of pixels with value greater than the threshold value is called bright region and its complement is called dark region. If the bright region has a higher fitness than the dark region, the bright region is the foreground. Otherwise, the dark region is the foreground. The foreground is the ROI extracted by this traditional algorithm. The threshold value plays a vital role in the ROI extraction and selecting an appropriate threshold value is the key to the success of this traditional ROI extraction algorithm. The performance of composite operators is compared with that of the traditional ROI extraction algorithm when the best threshold value is used. To find the best threshold value, every possible threshold value is tried by the algorithm and its performance is recorded.

#### 4.4.1. The traditional ROI extraction algorithm

1. find the maximum and minimum pixel values of the image.
2. if the maximum pixel value is greater than 1000
3. normalize the pixel values into the range of 0 to 1000. The pixel values are changed according to the following equation.

$$\text{new\_pixval} = (\text{org\_pixval} - \text{min\_pixval}) / (\mathit{max}\text{\_pixval} - \text{min\_pixval}) \times 1000$$

   where new_pixval and org_pixval are the new and original pixel values, respectively, and min_pixval and max_pixval are the minimum and maximum pixel values in the original image. After normalization, the minimum and maximum pixel values are 0 and 1000, respectively.
   else
4. do not normalize the image.
   endif
5. each integer value between the minimum and maximum pixel values is used as the threshold value and its performance in ROI extraction is recorded.
6. select the best threshold value and output its corresponding ROI.

Fig. 34 shows the ROIs extracted by the traditional ROI extraction algorithm corresponding to the best threshold value. The fitness values of the extracted ROIs and their corresponding threshold values are shown in Table 7. To extract ROIs from SAR and IR

Table 7
Fitness values of the extracted ROIs and the corresponding threshold values

| | Fitness | Threshold |
|---|---|---|
| Fig. 34(a) | 0.39 | 24 |
| Fig. 34(b) | 0.45 | 32 |
| Fig. 34(c) | 0.50 | 29 |
| Fig. 34(d) | 0.72 | 36 |
| Fig. 34(e) | 0.81 | 24 |
| Fig. 34(f) | 0.31 | 34 |
| Fig. 34(g) | 0.55 | 38 |
| Fig. 34(h) | 0.63 | 112 |
| Fig. 34(i) | 0.53 | 128 |
| Fig. 34(j) | 0.62 | 129 |
| Fig. 34(k) | 0.58 | 107 |
| Fig. 34(l) | 0.82 | 95 |
| Fig. 34(m) | 0.83 | 94 |
| Fig. 34(n) | 0.79 | 95 |
| Fig. 34(o) | 0.84 | 95 |
| Fig. 34(p) | 0.38 | 113 |
| Fig. 34(q) | 0.34 | 126 |
| Fig. 34(r) | 0.26 | 75 |
| Fig. 34(s) | 0.38 | 107 |

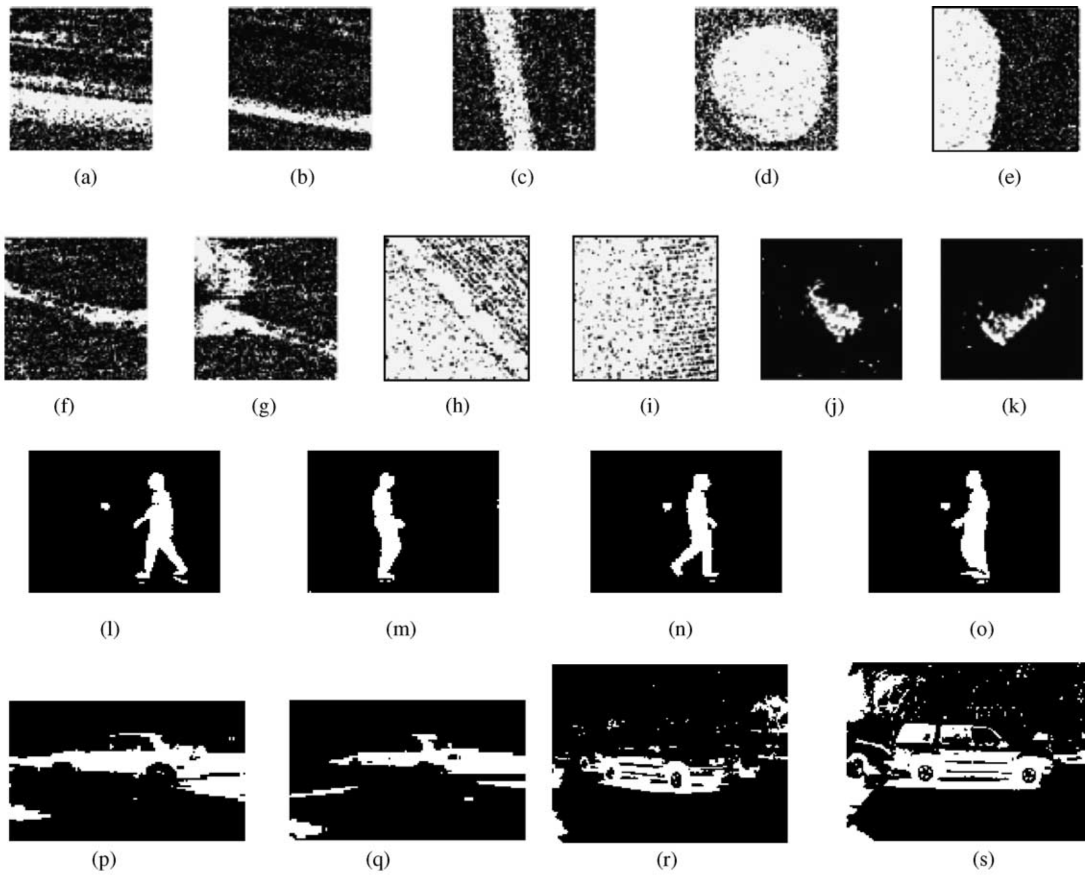

Fig. 34. ROI extracted by the traditional ROI extraction algorithm: (a) paved road vs. field; (b) unpaved road vs. field; (c) paved road vs. grass; (d) lake vs. field; (e) lake vs. grass; (f and g) river vs. field; (h and i) field vs. grass; (j and k) T72 tank; (l–o) person; (p and q) car; (r and s) SUV.

images, the original SAR and IR images are used by the traditional ROI extraction algorithm; to extract ROIs from RGB color images, the color images are first converted into gray intensity images and the traditional ROI extraction algorithm operates on the converted gray images. The value of a pixel in a converted gray scale image is the average value of RGB values of the corresponding pixel in the RGB color image. From Fig. 34 and Table 7, it is clear that the composite operators learned by GP are more effective in ROI extraction. Actually, its performance is better than the best performance of the traditional ROI extraction algorithm. Table 8 shows the average running time of the composite operators and the traditional ROI extraction algorithm in extracting ROIs from training and testing images. From Table 8, it is obvious that the composite operators are more efficient.

Table 8
Average running time (in seconds) of the composite operators and the traditional ROI extraction algorithm

| | Road | Lake | River | Field | Tank | Person | Car | SUV |
|---|---|---|---|---|---|---|---|---|
| Composite operator | 5 | 15 | 33 | 8 | 3 | 1 | 2 | 6 |
| Traditional ROI exaction algorithm | 38 | 25.5 | 68.5 | 37.5 | 26 | 4.8 | 5.5 | 20.5 |

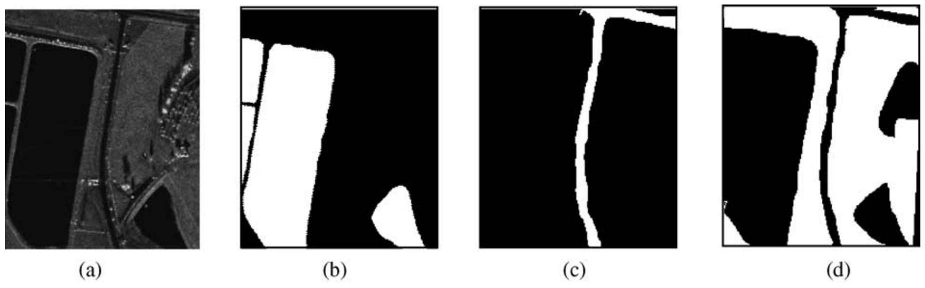

Fig. 35. SAR image containing lake, road, field, tree and shadow: (a) original image; (b) lake ground-truth; (c) road ground-truth; (d) field ground-truth.

### 4.5. A multi-class example

In the above examples, we showed the effectiveness and efficiency of composite operators learned by GP in ROI extraction. In this section, a complicated SAR image (shown in Fig. 35(a)) containing lake, road, field, tree and shadow is used as a testing image. Note that shadow is an unknown region (reject class, not considered in this paper) in this example. Fig. 35(b)–(d) show the ground-truth for lake, road and field.

We apply the composite operators for lake, road and field learned in examples 2,1 and 4, respectively, to the above testing image. The lake operator is applied first; then the road operator is applied to the rest of the image excluding the lake ROIs; finally, the field operator is applied to the rest of the image excluding both lake and road ROIs. The ROIs extracted are shown in Fig. 36. The fitness values are 0.85, 0 and 0.75, respectively. These results are not promising. Since the pixel values of road and lake regions are quite similar (see Fig. 37, many pixels in the road and lake regions have values between 0 and 20), the lake composite operator extracts part of the road and the road composite operator extracts no road pixel. In order to force GP to learn composite operators that can distinguish the subtle difference between the lake and road pixels, a SAR image (shown in Fig. 38) containing both lake and road is used as a training image. Fig. 38(a) shows the original image with the training regions from (4, 96) to (124, 119) and from (2, 25) to (127, 86) used by GP to learn composite operators for the lake extraction. To learn composite operators for the road extraction, the same image with the training region from (90, 30) to (135, 117) shown in Fig. 38(c) and the training image used in example 1 are used for training. Fig. 38(b) and (d) show the ground-truth for the lake and road, respectively. Note the images in Figs. 35(a) and 38(a) are quite different.

The steady-state GP is applied to synthesize composite operators for lake detection and the ROI extracted by the learned composite operator from the training image is shown in Fig. 39(a). The fitness value

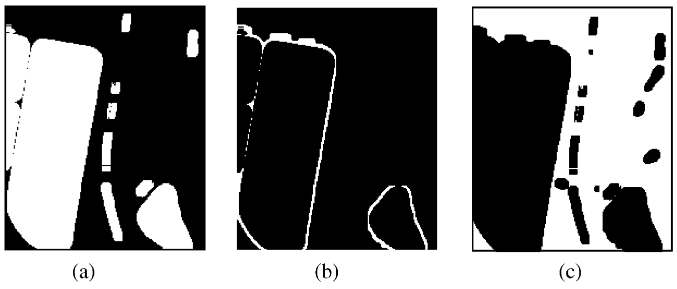

Fig. 36. Lake, road and field ROIs extracted by the composite operators learned in examples 2, 1 and 4: (a) lake ROI; (b) road ROI; (c) field ROI.

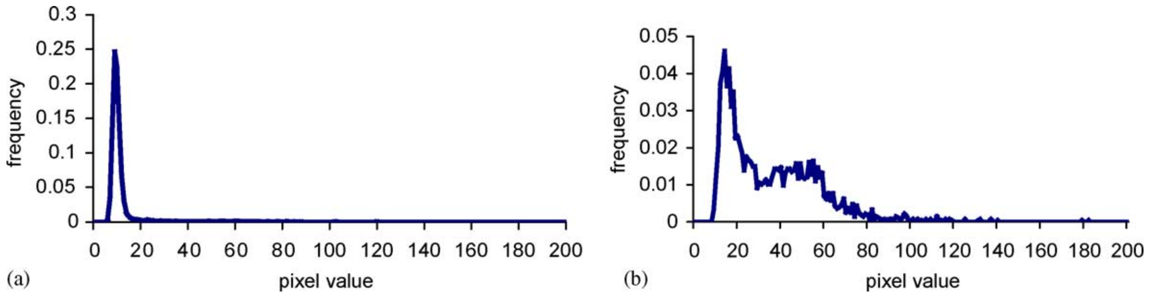

Fig. 37. Histograms of pixel values (range 0–200) within lake (a) and road (b) regions.

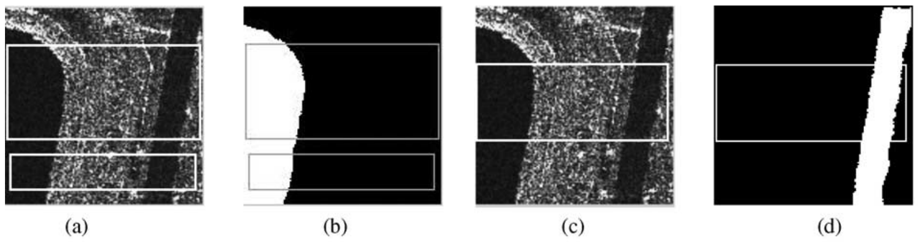

Fig. 38. SAR image containing lake and road: (a and c) original image; (b) lake ground-truth; (d) road ground-truth.

of the extracted ROI is 0.95. The generational GP is applied to synthesize composite operators for road detection. The ROIs extracted by the learned composite operator from the training images (Figs. 1(a) and 38(c)) are shown in Fig. 39(b) and (c). The fitness values are 0.78 and 0.90, respectively.

We apply the newly learned lake and road composite operators to the testing image in Fig. 35(a). The extracted lake and road ROIs are shown in Fig. 40(a) and (b). The fitness values of the extracted ROIs are 0.93 and 0.46, respectively. After extracting the lake and road from the image, we exclude the regions corresponding to the extracted lake and road ROIs and apply the field composite operator learned in example 4 to the rest of the image. The extracted ROI is shown in Fig. 40(c) and its fitness value is 0.81. The running times are 53, 127 and 26 s, respectively, for the results shown in Fig. 40.

Finally, the traditional ROI extraction algorithm is applied to the above testing image. The extracted ROIs, corresponding to the best threshold values, of the lake, road and field are shown in Fig. 41(a)–(c), respectively. To extract road, the regions corresponding to the extracted lake ROIs are removed and the algorithm is applied to the rest of the image. Fig. 41(b) demonstrates that it is very difficult for the traditional

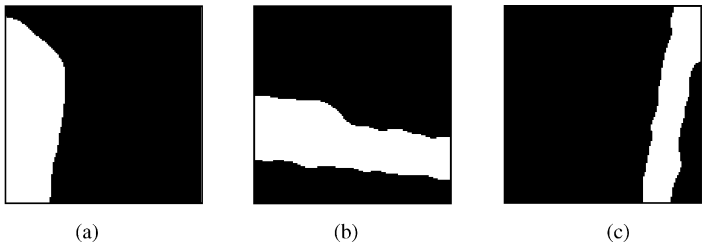

Fig. 39. lake and road ROIs extracted from training images: (a) lake ROI extracted; (b and c) road ROI extracted.

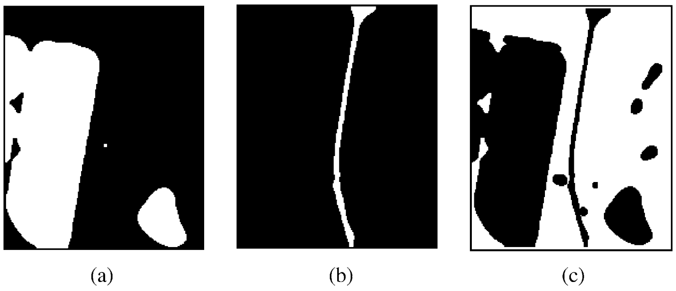

Fig. 40. Lake, road and field ROIs extracted from the testing image: (a) lake ROI; (b) road ROI; (c) field ROI.

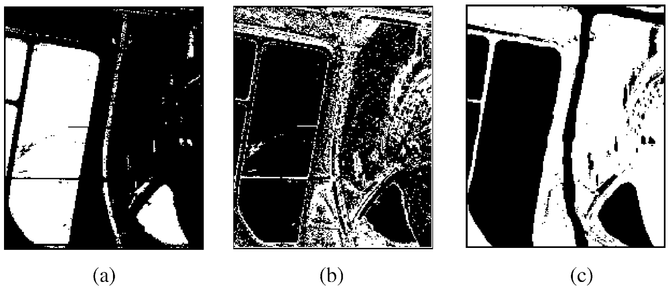

Fig. 41. Lake, road and field ROIs extracted by the traditional algorithm: (a) lake ROI; (b) road ROI; (c) field ROI.

ROI extraction algorithm to distinguish road from field in SAR images. To extract field, the regions corresponding to the ground-truth of the lake and road (not the ROIs corresponding to the lake and road) are excluded. The reason that we do not use extracted ROIs is that the extracted road ROIs are very bad. The fitness values of the extracted lake, road and field ROIs are 0.86, 0.15 and 0.79, respectively. The best threshold values are 16, 55 and 28.5, and the running times are 192, 176 and 195 s, respectively. It can be seen that the GP learned composite operators are more effective in the lake, road and field detection, compared to the traditional ROI extraction algorithm.

## 5. Conclusions

In this paper, we use genetic programming to synthesize composite operators and composite features to detect potential objects in images. We use soft composite operator size limit to avoid code-bloating and severe restriction on GP search. Our experimental results show that the primitive operators and primitive features defined by us are effective. GP can synthesize effective composite operators for object detection by running on the carefully selected training regions of training images and the synthesized composite operators can be applied to the whole training images and other similar testing images. We do not find significant difference between generational and steady-state genetic programming algorithms. GP has well known code-bloat problem. Controlling code-bloat due to the limited computational resources inevitably restricts the search efficiency of GP. How to reach the balance point between these two conflicting factors is critical in the implementation of GP. In the future, we plan to address this problem by designing new fitness functions based on the minimum description length (MDL) principle to incorporate the size of composite operators into the fitness evaluation process. Also, we will extend this work by discovering features within the regions-of-interest for automated object recognition.

## Acknowledgements

This research is supported by the grant F33615-99-C-1440. The contents of the information do not necessarily reflect the position or policy of the US government.

## References

[1] A. Ghosh, S. Tsutsui (Eds.), Advances in Evolutionary Computing—Theory and Application, Springer-Verlag, 2003.

[2] B. Bhanu, et al. (Eds.), IEEE Transactions on Image Processing—Special Issue on Automatic Target Recognition, vol. 6, no. 1, New York, USA, January 1997.

[3] J.R. Koza, Genetic Programming II: Automatic Discovery of Reusable Programs, MIT Press, 1994.

[4] C. Harris, B. Buxton, Evolving edge detectors with genetic programming, in: Proceedings of the First Annual Conference on Genetic Programming, MIT Press, Cambridge, MA, USA, 1996, pp. 309–314.

[5] R. Poli, Genetic programming for feature detection and image segmentation, in: T.C. Forgarty (Ed.), Evolutionary Computation, Springer-Verlag, Berlin, Germany, 1996, pp. 110–125.

[6] B. Bhanu, Y. Lin, Learning composite operators for object detection, in: Proceedings of the Conference on Genetic and Evolutionary Computation, NY, USA, July 2002, pp. 1003–1010.

[7] S.A. Stanhope, J.M. Daida, Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery, in: Proceeding of the Seventh Conference on Evolutionary Programming, Springer-Verlag, Berlin, Germany, 1998, pp. 735–744.

[8] D. Howard, S.C. Roberts, R. Brankin, Target detection in SAR imagery by genetic programming, in: Advances in Engineering Software, vol. 30, no. 5, Elsevier, May 1999, pp. 303–311.

[9] S.C. Roberts, D. Howard, Evolution of vehicle detectors for infrared line scan imagery, in: Proceedings of the First European Workshops on Evolutionary Image Analysis, Signal Processing and Telecommunications (EvoIASP'99 and EuroEcTel'99), Springer-Verlag, Berlin, Germany, 1999, pp. 110–125.

[10] W. Tackett, Genetic programming for feature discovery and image discrimination, in: Proceedings of 5th International Conference on Genetic Algorithm, Morgan Kaufmann, San Mateo, CA, USA, 1993, pp. 303–311.

[11] T. Belpaeme, Evolution of visual feature detectors, in: Proceedings of 1st Conference on Evolutionary Computation in Image Analysis and Signal Processing, Göteburg, Sweden, 1999, pp. 1–10.

[12] M. Koeppen, B. Nickolay, Genetic programming based texture filtering framework, in: Pattern Recognition in Soft Computing Paradigm, World Scientific, 2001, Chapter 12, pp. 275–305.

[13] M. Johnson, P. Maes, T. Darrell, Evolving visual routines, Artif. Life 1 (1994) 4.

[14] S. Ullman, Visual routines, Cognition 18 (1984) 97–159.

[15] B. Bhanu, S. Fonder, Learning-integrated interactive image segmentation, in: A. Ghosh, S. Tsutsui (Eds.), Advances in Evolutionary Computing—Theory and Application, Springer-Verlag, 2003, pp. 863–895.