PERGAMON

Available at www.ElsevierComputerScience.com POWERED BY SCIENCE DIRECT®

Pattern Recognition 37 (2004) 61 – 77

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Functional template-based SAR image segmentation

Bir Bhanu*, Stephanie Fonder

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

Received 22 October 2002; accepted 3 June 2003

Abstract

We present an approach to automatic image segmentation, in which user selected sets of examples and counter-examples supply information about the specific segmentation problem. In our approach, image segmentation is guided by a genetic algorithm which learns the appropriate subset and spatial combination of a collection of discriminating functions, associated with image features. The genetic algorithm encodes discriminating functions into a functional template representation, which can be applied to the input image to produce a candidate segmentation. The performance of each candidate segmentation is evaluated within the genetic algorithm, by a comparison to two physics-based techniques for region growing and edge detection. Through the process of segmentation, evaluation, and recombination, the genetic algorithm optimizes functional template design efficiently. Results are presented on real synthetic aperture radar (SAR) imagery of varying complexity.
© 2003 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: SAR images; Segmentation evaluation; Template design; User-directed image segmentation

1. Introduction

The segmentation problem involves partitioning an image into regions which are homogeneous within themselves and distinct from each other, according to some set of criteria. There are a variety of approaches to image segmentation, including edge detection, region splitting/merging and clustering-based techniques. Each of these approaches suffers from sensitivity to parameters for thresholding, and/or termination conditions. Still other approaches combine a few of these methods. However, the underlying cause for many of these algorithms to perform poorly is the inability to specify how homogeneous a region should be and how distinct bordering regions should be in an application dependent manner.

A simplified system diagram of the approach presented in this paper is given in Fig. 1. The application dependency is overcome by allowing the user to interactively train the segmentation tool for their own application. The image segmentation is guided by a genetic algorithm (GA) which learns the appropriate subset and spatial combination of a collection of functions, associated with image features, designed to discriminate one image region from another. In an interactive session, the user selects a set of examples and a set of counter-examples. The input SAR image is 'denoised' to reduce the speckle effect and then edge detection and region growing (with example regions as the seed) are performed on the denoised result. The example and counter-example sets are used to scale the data and create histograms, which quantify the class separation for a variety of features. Based on histogram overlap, a set of discriminating functions is designed to perform discrimination between the example class and counter-example class.

A GA encodes these functions into a functional template representation and produces a population of initial functional templates. These functional templates are applied to the input image to produce segmentations. The results of the segmentations are evaluated using a fitness function, and the population of functional templates are combined and modified via a set of operations based on genetic evolution, in an effort to evolve an optimized segmentation agent. The

* Corresponding author. Tel.: +1-909-787-3954; fax: +1-909-787-3188.
E-mail addresses: bhanu@cris.ucr.edu (B. Bhanu), steph@cris.ucr.edu (S. Fonder).

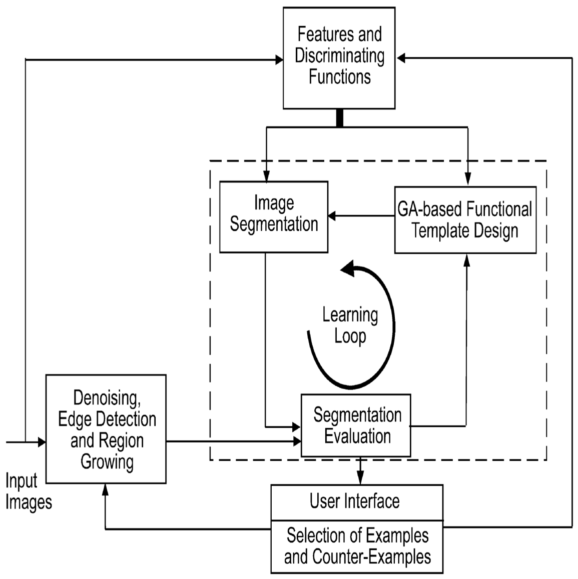

Features and Discriminating Functions
Image Segmentation
GA-based Functional Template Design
Learning Loop
Denoising, Edge Detection and Region Growing
Segmentation Evaluation
Input Images
User Interface
Selection of Examples and Counter-Examples

Fig. 1. System for learning-integrated interactive segmentation.

process of segmentation, evaluation, and recombination, is repeated for a given number of generations and the best result of the final generation (after post processing) is presented as output to the user. Although the GA evaluates candidate segmentations via a comparison to region- and edge-based techniques, these are only used as a guide for the process of searching through possible segmentation outcomes produced by the combinations of the discriminating functions and their spatial arrangement. Since the discrimination functions inherently contain classification information, it is possible to outperform the region- and edge-based approaches that are used to evaluate the segmentation quality.

In the following Section 2, we provide background, related work and the contributions of this paper. Section 3 provides the details of the technical approach. Section 4 presents experimental results and Section 5 presents the conclusions of this paper.

2. Background, related work and contributions

Learning-based approaches have been explored to allow segmentation to adapt to the appropriate problem domain as well as incorporate new information about a given domain quickly [1–4]. Bhanu and Lee [5] examine numerous function optimization techniques including GAs, simulated annealing and hybrid techniques such as the combination of GAs and hill climbing. GAs suffer from potential premature convergence and computationally expensive segmentation evaluation. Since simulated annealing relies on slow 'cooling' to avoid local optima, it is inherently a slow process. Neural networks can approximate Bayesian performance with appropriate design, however, the training process is often complex and design solutions are typically not scalable. Further, they need large amounts of training data, which are often unavailable.

In the context of image segmentation, classification is the problem of assigning meaningful labels to regions. An equivalent formulation of the classification problem is assigning a label to each pixel, regions are then simply the connected components of identical labels. Thus, in this formulation segmentation and classification are performed simultaneously. Some typical pattern recognition techniques applied to the classification problem are Bayesian [6], K-nearest neighbor [7], and template matching [8] methods.

A *traditional template* classifier applied to images specifies a function of local intensities, typically represented as a matrix of values. The template is correlated with the image and the results are thresholded to produce a classification result. *Functional templates* modify this technique, such that each element of the matrix is an index to a function. The advantage of functional templates over standard templates is the potential for better discrimination, which may occur when indexed functions are nonlinear. These functions can also encode sensor-specific and/or class-specific information into the functional template. Furthermore, it provides a framework to combine information from multiple features to potentially increase the discrimination power over segmentation approaches based on a single feature. The drawback of this technique is the need for functional template design; that is, the selection of which functions to incorporate and where to place them within the functional template, which is difficult due to the complex interactions taking place between various features and associated functions.

The work presented in this paper is an improvement over previous functional template work. Large size (order of 20×50) aspect dependent functional templates used for object recognition were manually designed in Ref. [9], which was a very time consuming process. In Ref. [8], functional template design was automated, by limiting the design space to simple single feature templates, where the selected feature minimized Bayesian risk. The focus of our research is the development of an approach which automates functional template design that allows for multiple functions and evaluate its performance for the segmentation of SAR images.

2.1. SAR image segmentation

Algorithms developed for other types of imagery are not directly applicable to SAR, because of the differences in image properties. Although there is literature which performs learning-based segmentation and/or classification, there are presently no interactive approaches such as the approach presented in this paper [10]. Furthermore, all SAR learning-based segmentation and/or classification related work approaches the problem by producing a segmentation and classifying the resulting regions rather than performing pixel-level classification as presented here. Li et al. [11] assign classes to clusters using the distance from class means obtained from training data. Baraldi and Parmiggiani [12] use a neural network to cluster data using parameters calculated on initial segments. Classification of clusters is then accomplished via domain dependent knowledge-based classification scheme. Learning-based segmentation has also been explored with no attempt at classification of the resulting regions. Most learning-based segmentation attempts use neural networks [13–15]; however, Gou and Ma [16] present a clustering method which uses entropy-based threshold to break the feature space into 'mode' and 'valley' regions. These 'mode' regions are further processed to produce a final segmentation. Still other approaches perform segmentation without learning, then apply learning-based classification techniques to the segmented regions. Shoemakers et al. [17] use an edge detection and region growing hybrid for segmentation and classification of the resulting regions with a neural network. Gagnon and Klepko [18] also use neural networks for classification but use directional thresholding and region growing for segmentation. Soh and Tsatsoulis [19,20] perform segmentation using dynamic local thresholding, but use clustering to perform classification of regions. Rogers et al. [21] use various neural networks based approaches for preprocessing and segmentation.

2.2. *Genetic algorithms and image segmentation/classification*

GAs are a learning technique based on biological evolution. Given a population of candidate solutions, a fitness function is used to evaluate each individual of the population. Each generation the individuals (i.e., candidate solutions) are evaluated and recombined producing the next generation's population. Recombination is designed to allow fit individuals to pass on important characteristics to the next generation. Because the nature of genetic optimization is randomized search from a number of search points (the individuals), the approach is ideal for parallel implementation. GAs have been applied to optimize overall segmentation quality [1,5,12,22,23], segmentation parameter selection [4,5,24–27] and to feature design for pixel-level classification [28–30].

GAs have also been used to determine the optimal subset of features for pixel-level classification. The approach presented in this paper is one example of such work. Matsui et al. [31] use an offline GA to select the optimal combination of feature indices for tissue classification without testing the neural network classifiers into which features are incorporated. Campbell and Thomas [32] use an offline GA to select a subset of Gabor filters requiring fewer convolutions for the classification technique in which they are eventually incorporated. Erdogan et al. [33] extract optimal texture features from several co-occurrence matrix based texture measures and training determines optimum ranges of the measures on known texture regions.

2.3. *Contributions of this paper*

Unlike previous work on functional templates [9,34,35], in this paper we design functional templates in a systematic manner using GAs. Besides the selection of a subset of features and associated functions, the GA determines the spatial placement of functions within the template. During the learning process, segmentation evaluation is performed by a comparison of candidate segmentation to two segmentation techniques: edge- and region-based. These are physics-based techniques that incorporate SAR-specific information to produce a segmentation using a log likelihood ratio test where the distributions used in the test are specifically developed for SAR imagery. The novel fitness function, compares a candidate segmentation to portions of the physics-based region and edge estimates using two terms. The region term encourages a segmentation to correctly classify pixels within the region from which the examples were selected. The edge term encourages regions classified as example regions to have edges coinciding with image edges. A prototype of the system has been developed and tested on SAR imagery. Our experimental results show that genetically designed functional templates perform better than functional templates designed using the Bayesian best feature.

3. Technical approach

Fig. 1 presents an overview of the approach to image segmentation. The input to the system is an image. Since the system is designed for segmentation of SAR imagery, the input image is denoised. While the system is denoising the image and features are calculated, an interactive session with the user occurs, where example and counter-example sets of pixels are selected. The examples provide input to a region growing process performed on the denoised image, whose results in combination with edge-detection results are used to perform segmentation evaluation within the GA. In addition, examples and counter-examples are used to create a discriminating function for each feature. The features, represented by their corresponding discriminating functions, are then incorporated into functional template design, which is optimized over successive generations of the GA. When the GA terminates, results are presented back to the user after postprocessing.

3.1. *Pre-learning phase*

This subsection discusses the aspects of the approach which occur before or after the GA attempts to optimize the functional template design in the learning phase. The pre-learning components of the approach include user-interaction, wavelet denoising, computation of physics-based segmentations for evaluation, feature calculation, discriminating function design and calculation of associated weights.

3.1.1. User-interaction

During the user-interaction the image is simply displayed and an example set and a counter-example set are selected by the user. Since, the examples represent class ω_1 and the counter-examples represent any other classes present in the data, the approach is inherently a two-class segmentation/classification (which is extended to N-class segmentation in Section 4.2). Once the system presents results to the user, the following things can happen: (a) the user accepts the results on the test image. (b) the user selects different examples and counterexamples. For the results presented here, the user selects only one example and one counter-example. This user-interaction in the form of examples and counter-examples is the reason for the application independent nature of the approach because the user is allowed to train the segmentation system for the application at hand.

3.1.2. Wavelet denoising

Wavelet denoising is applied to dB SAR imagery to remove additive noise without loss of spatial resolution. The procedure for speckle reduction depends on the initial shift of the signal. However, a shift invariant result is obtained by averaging the results for all possible shifts of the input signal (as described by Odegard et al. [36]).

3.1.3. Physics-based segmentation for evaluation

The set of examples is input to a region growing algorithm whose results are used later, with the results of an edge-detection algorithm, in evaluating the fitness or quality of candidate segmentation results in the learning phase of the approach. However, the region growing and edge-detection results can be calculated before the learning phase, so that only the comparison between these results and the candidate solution need be computed while the GA is learning.

A physics-based edge-detection test and two physics-based segmentation algorithms (which use the test) are presented below, all of which incorporate the SAR speckle model. Both segmentation algorithms are based on the cartoon model [37], which states that images are made up of regions, which are separated by edges, and that regions are homogeneous according to some criterion. This criterion is assumed to be radar cross-section (RCS, denoted as σ°). Because regions are separated by edges according to this model, a test which determines whether an edge is present can be used for both the region growing and edge detection algorithms.

While we implement a 2-D edge-detection algorithm, for simplicity, the following discussion is based on a 1-D version [37]. A maximum likelihood test for detecting edges based on a model of SAR speckle states

$$P(I) = \prod_{k=1}^{M} 1/\sigma^\circ \exp[-I_k/\sigma^\circ].$$

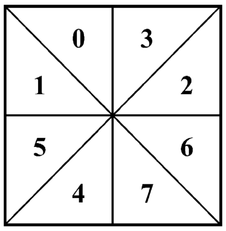


Fig. 2. Edge detection algorithm subwindows for detecting edges every 45°.

The test determines whether an edge is present at position k in a one-dimensional window containing M pixels with intensities $I_1, I_2, \ldots, I_k, \ldots, I_M$.

The window is assumed to contain either one or two regions. If the window contains two regions, A and B, their respective RCS are σ_A° and σ_B°, otherwise $\sigma_0^\circ = \sigma_A^\circ = \sigma_B^\circ$. Given the speckle model, the joint probability of such a window is

$$P_{A,B}(\sigma_A^\circ, \sigma_B^\circ | I_j, k) = \left(\prod_{j=1}^{k} 1/\sigma_A^\circ \exp - I_j/\sigma_A^\circ \right) \times \left(\prod_{j=k+1}^{M} 1/\sigma_B^\circ \exp - I_j/\sigma_B^\circ \right).$$

$\bar{I}_X$ estimates the RCS of a region X as the mean of the portion of the region contained in the window, where lower subscript o refers to the entire window. Given this, the log likelihood estimate of detecting an edge is reduced to

$$\lambda_D(k) = -k \ln \bar{I}_A - (M-k) \ln \bar{I}_B + M \ln \bar{I}_0.$$

The sensitivity of this test is based on the threshold, which can be chosen according to false alarm probabilities derived from the distribution of $\bar{I}_A/\bar{I}_B$. Since the value of this ratio is ideally 1, if only one region is represented in the window, thresholds t_2 and t_1 must be chosen above and below 1, respectively. (Edges occur where $\lambda_D \notin [t_2, t_1]$.) These thresholds can be related to λ_D via the following two equations:

$$\lambda_D(k) = -k \ln t_1 - M \ln M + M \ln[M - k + kt_1], \quad t_1 < 1,$$

$$\lambda_D(k) = k \ln t_2 - M \ln M + M \ln[M - k + k/t_2], \quad t_2 > 1.$$

3.1.4. Edge-detection

The 2-D edge detection approach is a computationally efficient extension of the 1-D method described above that is designed to smooth the RCS estimates and detect edges at 45° increments in a single pass. Each orientation divides a window into two regions, A and B, each with 4 adjacent sub-windows as shown in Fig. 2.

The following pseudocode describes the algorithm:

```
for each pixel{
  calculate intensity subtotals for each of the eight subwindows.
  for each orientation{
    Compute region A and region B means
      (left and right side of the assumed orientations) from the
      eight subtotals.
    Compute edge magnitude.
  }
  Record max edge magnitude and the orientation which
produced it.
}
pixels with edge magnitudes in the top 20% are marked as edges.
```

A 12×12 window is used to obtain reliable estimates of the RCS. Because of this, the algorithm produces very thick edges which are thinned by unmarking edge pixels whose edge magnitude is not the local maximum of its 8-neighbors, reducing edges to a single pixel width. When used with measures such as edge–border coincidence for fitness computation, the region boundary would have to match thinned edges exactly. In order to provide more flexibility to segmentation evaluation measures, the edges are thickened to a width of 3 pixels, by marking the pixel on either side of each thin edge pixel.

3.1.5. Region growing

The region growing algorithm starts with the example pixels as a seed. It uses 5×5 blocks in order to find a reliable RCS estimate for the region the block is in. There are two thresholds, above and below the value one, which are used by the edge test to determine whether to merge the blocks. Since a result that 'overgrows' the region would misguide the fitness function, conservative thresholds are empirically determined. The following pseudocode describes the region growing algorithm.

```
while (merged>0){
   set merged = 0
   for each $5 \times 5$ block around the perimeter of the current
grown region{
      Apply the edge test to compare the $5 \times 5$ block and the
grown region
      if (0.8 <= edge magnitude <= 1.2){
         merge block into grown region
         set merged=merged+1;
      }
   }
}
```

3.1.6. Features

Table 1 lists the features used in this work. The index associated with a given feature is used consistently to refer to the discriminating function based on that feature in functional templates. The first seven features represent local image intensity statistics, while the remainder specify means or standard deviations of Gabor wavelet filtered images. Two scales (13×13 and 11×11) and four orientations ($0°, 45°, 90°, 135°$) of Gabor wavelet filters are used and then means and standard deviations are calculated from a 13×13 area rather than the entire image in order to have a feature value at every pixel. The 13×13 local area was chosen empirically, because it provided a better estimate than smaller areas while still providing fluctuation from pixel to pixel. Also, areas larger than this caused biased feature values on larger areas of the image boundary.

3.1.7. Discriminating functions and weights

Given the example and counter-example sets as well as the feature images, a discriminating function and a weight are calculated for each feature. The design of the function is automated by a set of equations which measure the overlap of example and counter-examples in the feature value histogram. The weight associated with this discriminating function is based on the Bayesian classification error of the feature. Before calculating discriminating functions, feature values are scaled to the range $[0, 255]$ by mapping the mean μ to 128 and $\mu \pm 3\sigma$ (σ is standard deviation) to 0 and 255, respectively.

Table 1
Features used and their corresponding indexes

Index	Feature
0	Intensity
1	3×3 local mean
2	3×3 local standard deviation
3	5×5 local mean
4	5×5 local standard deviation
5	7×7 local mean
6	7×7 local standard deviation
7	Scale: Large, Orientation: 0° mean
8	Scale: Large, Orientation: 45° mean
9	Scale: Large, Orientation: 90° mean
10	Scale: Large, Orientation: 135° mean
11	Scale: Small, Orientation: 0° mean
12	Scale: Small, Orientation: 45° mean
13	Scale: Small, Orientation: 90° mean
14	Scale: Small, Orientation: 135° mean
15	Scale: Large, Orientation: 0° standard deviation
16	Scale: Large, Orientation: 45° standard deviation
17	Scale: Large, Orientation: 90° standard deviation
18	Scale: Large, Orientation: 135° standard deviation
19	Scale: Small, Orientation: 0° standard deviation
20	Scale: Small, Orientation: 45° standard deviation
21	Scale: Small, Orientation: 90° standard deviation
22	Scale: Small, Orientation: 135° standard deviation

The histogram of examples and counter-examples has 255 bins, where a feature which discriminates well should have example pixels near the mean and most counter-examples at the extreme values. The following function, F_l (patterned after [33]) is then applied to each bin, l, of the example and counterexample histogram, denoted by E_l and C_l, respectively, to produce a score for that feature value. Bins with many examples and few counter-examples produce high scores, while the opposite case produces low scores. The function $H(.)$, clips values to the range $[0, 1]$. The parameters of $H(.)$ determine histogram overlap, while the 16 and -16 factors scale the result. Special cases are handled when there are no examples, no counter-examples, or both. The discriminating function is given by

$$F_l =$$

$$\begin{cases} 0.0 & \text{if } E_l = 0 \text{ and } C_l = 0, \\ 16 \times H\left(4 \times \left(E_l / \sum_{l=0}^{255} E_l\right)\right) & \text{if } C_l = 0, \\ -16 \times H\left(4 \times \left(C_l / \sum_{l=0}^{255} C_l\right)\right) & \text{if } E_l = 0, \\ 16 \times H((1/39) \times ((E_l/C_l) - 1)) & \text{if } E_l > C_l, \\ -16 \times H((1/39) \times ((C_l/E_l) - 1)) & \text{otherwise.} \end{cases}$$

The Bayesian weight, W_l, associated with the discriminating function is defined as

$$W_l = 1.0 - \left(2 \times \sum_{l=0}^{255} \frac{\min(C_l, E_l)}{(C_l + E_l)}\right).$$

3.2. Learning loop

The learning loop consists of image segmentation, segmentation evaluation and evolutionary template design processes that use a GA approach for optimization. Although the design decisions such as representation, operator design, parameter selection, and fitness function design are done before the learning phase, they are discussed here because of their relevance to the GA.

The design of the functional template of a given size requires a solution of a combinatorics problem. For example, for 20 functions, the size of the search space for a 3×3 template is 512 billion. We use GAs as function optimizers [5] for functional template design since they allow the possibility of achieving the global maximum without exhaustive search. The appropriate combination and arrangement of discriminating functions are optimized by the GA for good segmentation results. A GA is composed of a population of candidate solutions, or individuals. In this research, an individual is a functional template, represented as a 2-D array of binary strings. The binary to decimal conversion of each bit string gives the index of the discriminating function represented for that location in the functional template. Each discriminating function is developed from a single feature, and referenced by the index associated with that feature. Features and their indexes are given in Table 1. Size large and small refer to 13×13 and 11×11, respectively. Features 0 to 6 are intensity based features, and features 7 to 22 are Gabor wavelet based features.

3.2.1. Segmentation

After randomly generating generation zero, the initial population of the GA, the segmentation associated with each functional template is then produced using the functional template classification rule:

$$\sum_{i=-M/2}^{M/2} \sum_{j=-N/2}^{N/2} W_{i,j} \times S_{i,j}(I(a+i, b+j)) > t$$

Assign $I(a, b)$ to class ω_1,

$$\sum_{i=-M/2}^{M/2} \sum_{j=-N/2}^{N/2} W_{i,j} \times S_{i,j}(I(a+i, b+j)) \leqslant t$$

Assign $I(a, b)$ to class ω_2.

The above equation describes the classification of an image pixel $I(a, b)$ using a $(M + 1) \times (N + 1)$ functional template T. $S_{i,j}$ denotes the discriminating function F_l indexed at position $T(i, j)$ of the template and $W_{i,j}$ is the weight (W_l)

associated with function $S_{i,j}$. In this paper, all templates are 3×3 and the threshold $t = 0$.

After producing a segmentation for each individual, the segmentations are evaluated by the fitness function described in the next subsection.

3.2.2. Segmentation evaluation and fitness function

Once an image segmentation result on an image has been obtained, we need to evaluate its quality using a fitness function. In real applications groundtruth information is generally not available. In our approach, all that is known about the image is that the user selected example set is part of a region corresponding to class ω_1 and the user selected counter-example set is part of a region corresponding to class ω_2. The fitness function is made up of two terms, a region term, $T1$, and an edge term, $T2$.

The intuition for the region term of the fitness function is the desire to fill in as much of the example region as possible. The region term uses the grown region as the groundtruth region. Thus, if R is the set of pixels in the segmented example region, a candidate segmentation (the region incorporating the example set) and G is the set of pixels from the grown region

$$T1 = n(G \cap R)/n(G),$$

where $n(X)$ is the number of pixels in set X.

If only the region term were used for fitness, a segmentation which classified all pixels as ω_1 would receive the same value as a segmentation that classified all pixels correctly. However, the fitness function should not prevent other regions in the image from being classified as belonging to class ω_1, since other regions pertaining to class ω_1 may be contained in the input image. Intuitively, the second fitness term should prevent evolved segmentations from having regions which ‘overrun’ their region boundaries. The edge term is defined below, where S is the boundary of the region in the segmentation image and E is the set of detected edges within the minimum bounding rectangle of the region growing result, extended 10 pixels in each direction. This is essentially edge border coincidence (EBC) within a limited area of the image, multiplied by a factor of 3. Because edge detection first thins edges to 1 pixel and then dilates them again to a width of 3 pixels, $n(E)$ is typically 3 times as large as a good segmentation boundary. So, edge border coincidence is multiplied by 3 to account for this effect. Thus, edge term, $T2$, is defined as

$$T2 = 3 \times n(E \cap S)/n(E).$$

Combining the region term and the edge term, the overall fitness is then defined as

$$fitness\ function = (T1 + T2)/4.$$

3.2.3. Evolutionary template design

Having evaluated every individual in a generation N, generation $N + 1$ is obtained through a recombination process consisting of the application of three genetic operators: selection, crossover, and mutation. Selection is applied twice, to select high fitness individuals for input to the crossover operator, which swaps discriminating function indexes between the selected individuals, the parents. The output of each application of the crossover operator are two new individuals, the children, who replace low fitness individuals from the population, also chosen by the selection operator. After crossover, the mutation operator is applied to the entire population. Evolution is completed when the termination criteria are met.

Selection: A standard tournament selection method was used. Tournament selection is the random selection of a subset (of a certain size) from the population. From this subset, the individual with the highest fitness or lowest fitness is chosen when selecting for crossover or replacement (i.e., death), respectively.

Crossover: Crossover is the combination of two individuals to produce two new individuals and its rate is the percentage of the population to participate in crossover each generation. A crossover operator that preserves the 2-D spatial information has been developed, based on the hypotheses that spatial layout of information in the template is important. Crossover is performed by randomly selecting two locations in the functional template to define a rectangle. The first location corresponds to the upper right corner of a rectangle and the second to the lower left. Copies of the rectangle are placed on both parent functional templates, and elements within the rectangle boundary are swapped. In our crossover method bias is removed by allowing the selected rectangle to conceptually wrap around in both horizontal and vertical directions (this makes the probability of a given element to be swapped equal for every element in the functional template). This crossover operation is illustrated in Fig. 3 where the selected locations are marked on parent 1, while the rectangle resulting from this selection is indicated on parent 2.

Mutation: For mutation, the standard operator was chosen. That is, each bit in the representation of the functional template is flipped (i.e. mutated) with probability p_μ, where p_μ is the rate of mutation.

The final output of the GA, the evolved functional template is the functional template with the highest fitness in the final generation.

3.3. Post-processing

Post-processing is performed after the GA terminates and consists of two tasks: *false positive removal* and *false negative filling*. False positives occur when pixels belonging to the counter-example class are incorrectly classified as belonging to the example class. *False positive removal* finds the connected components using the segmentation results and removes any component making up less than $X\%$ (default: $X = 1$) of the image. False negatives occur when pixels belonging to the example class are classified as

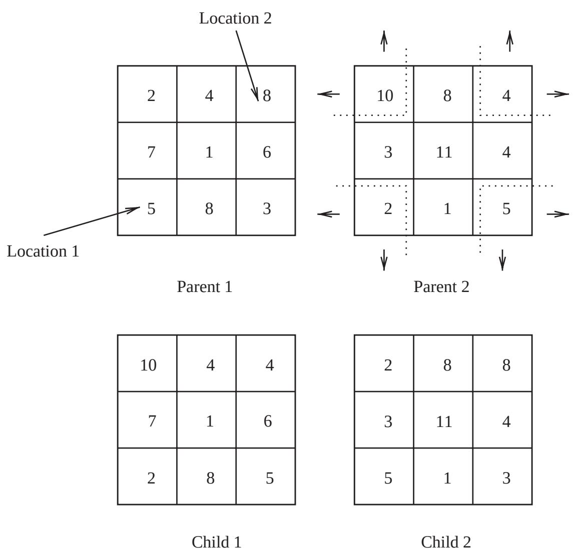


Fig. 3. Illustration of a crossover operator incorporating 2-D wrap around.

belonging to the counter-example class. *False negative filling* looks at each pixel which is unmarked, if Y (default: $Y = 6$) or more of its 8-neighbors are marked with the identical class labels the pixel is assigned this class label.

N-class post-processing consists of an additional processing step. N-class classification is actually performed as $N+1$ class classification, where the $N + 1$th class is an unknown class consisting of pixels not belonging to classes 1 through N. If there are known to be only N classes present in the data, *majority filtering* is performed to reclassify pixels which were originally classified to class $N + 1$. These pixels are reclassified as the class that the majority of its neighbors were assigned to.

4. Performance results and analysis

Several experiments are performed using real SAR images to demonstrate the performance of the technique. They include two-class and multiple class experiments. Also experiments are shown where the evolved functional templates obtained during training are applied to a new image. Further, experiments are performed that demonstrate the effectiveness of the crossover operator. For all of these experiments the value of parameters used in the algorithms are kept the same. These parameter settings are: template size 3×3, population size 100, tournament size 10, crossover rate 0.25 and mutation rate 0.01, number of generations$=$10. The edge detection picks up the important edges, but lots of extra edges as well. Fortunately, most are outside of ± 10 pixels of region growing estimate and these are the only edge pixels used by the fitness function.

Several measures of segmentation quality are presented for experiments which demonstrate the quality of evolved segmentations. The fitness function and its individual components are used for both evaluation and interpretation of results. The fitness of an individual demonstrates the evolutionary performance, while the terms of the fitness function add intuition about their relative importance and interaction. All of the additional measures use groundtruth information and are not used during evolution, but are useful for final quantification of results.

The percentage of Pixels Classified Correctly (PCC) is defined as

$$PCC = \max(1 - [(n(G) - n(G \cap R)) + (n(R) - n(G \cap R))/n(G)], 0),$$

where $n(X)$ is the number of pixels in the set X, G is the set of pixels which belong to class ω_1 from the groundtruth image, and R is the set of pixels classified as belonging to class ω_1 in the segmented image. PCC measures the overall performance, but can sometimes be misleading. This is particularly true when there are many more counter-example pixels than example pixels or vice-versa. As this is often the case, it gave rise to three additional measures, example accuracy (EA), counter-example accuracy, and normalized percent pixels classified correctly (NPCC). These measures are defined below where R, G, and $n(X)$ are defined as above and X' is the complement of X. NPCC is particularly useful as it gives more meaningful results in the cases where PCC is misleading, because it is normalized by the class populations

$$EA = n(R)/n(G),$$

$$CA = n(R')/n(G'),$$

$$NPCC = (EA + CA)/2.$$

EA is of most concern since templates are trained to discriminate the example class in later testing images. However, EA does not penalize for segmentations which ‘overrun’ the example region and include many counter-examples. Thus, both EA and CA must be considered for analysis.

Evolved results are compared to three default segmentations (Bayesian, PCC, and NPCC), which use a single feature in all functional template locations. The Bayesian default selects the (discriminating function) feature which minimizes classification error for the example and counter-example sets. (In our discussion in this section we use the word feature and discriminating function interchangeably.) However, the remaining two defaults utilize groundtruth. Single feature templates are exhaustively applied to the image and the segmentation corresponding to the feature which maximizes PCC is selected for the PCC default. Similarly, the segmentation corresponding to the feature which maximizes NPCC is selected for the NPCC default. Often two or more of the defaults select the same feature, and thus correspond to the same segmentation.

All SAR data in these experiments are obtained from the MSTAR (public) clutter data set. All images are one foot resolution X-band data at a 15° depression angle. In the experiments, borders of examples are marked in yellow and blue, and border of counter-examples are marked in green and purple.

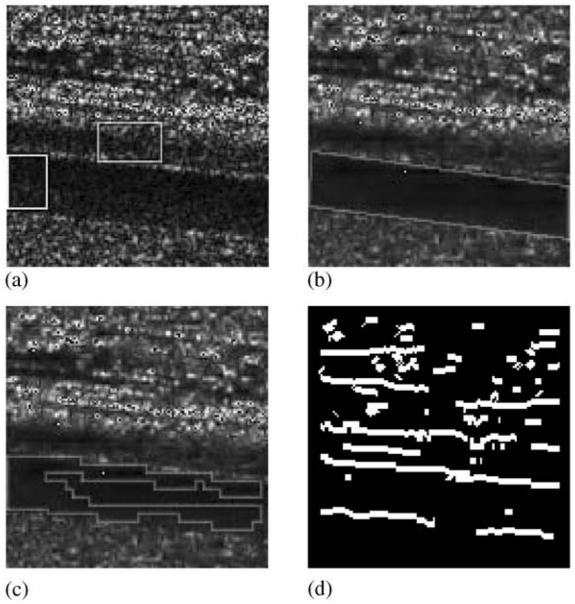


Fig. 4. Paved road vs. field: (a) original image with examples (yellow/blue) and counter-examples (green/purple), (b) denoised image with ground truth (red), (c) region growing (Red) and (d) edge results.

4.1. Two-class image segmentation

This subsection provides details on three typical experiments and a discussion on the overall two-class results.

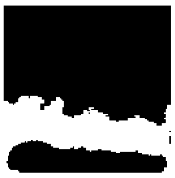
4.1.1. Example 1: paved road vs. field

Fig. 4 shows the original image, results of denoising, region growing, and edge detection, as well as ground truth for the image with paved road and field. Fig. 5 presents the overall results which include segmentation results and the corresponding templates. Feature 8 is the best Bayesian feature (which also happens to be the best feature for NPCC), feature 5 is the best feature for PCC. The final evolved results are better than the best Bayesian and the best PCC default, both according to PCC and NPCC.

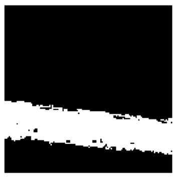
In the evolved results, every template has 3 or 4 instances of the local intensity features, involving some combination of intensity and 3×3 mean. It is apparent from the PCC default that intensity features do much of the classification; however, the boundary and some gaps inside the example area are compensated for by other features, such as local intensity standard deviation and Gabor standard deviation.

4.1.2. Example 2: paved road vs. grass

The region growing and edge detection results for paved road vs. grass are given in Fig. 6 along with original and denoised images, example/counter-example sets and ground truth. Region growing finds just over half of the road, as the region only grew downward. Edge detection finds the edges of the road particularly well, as well as some texture edges in the grass class. Although a few of the extra texture edges are included in the fitness term, the majority are excluded. The evolved segmentation results in Fig. 7


Bayesian/NPCC Default

PCC Default


GA Result

8	8	8
8	8	8
8	8	8

5	5	5
5	5	5
5	5	5

0	1	0
19	19	2
13	13	16

Evaluation Quantities	Bayes	PCC	Best GA Result	Ave. GA Result
Region Term	0.837	0.866	0.869	0.881
Edge Term	0.300	0.458	0.533	0.642
Fitness Function	0.285	0.331	0.351	0.254
EA	0.906	0.872	0.916	0.912
CA	0.944	0.965	0.991	0.970
PCC	0.936	0.945	0.975	0.957
NPCC	0.925	0.919	0.954	0.941

Fig. 5. Paved road vs. field results.

are better than the default results. It is not surprising then, that evolved results have significantly better example accuracy performance and have nearly perfect counter-example accuracy, which is consistent with the defaults. Thus, the evolved results are significantly better than default segmentations according to the NPCC. (The PCC does not show this improvement as dramatically since example pixels represent only 23% of the image.) The intensity feature was the most common in the evolved results (Fig. 7). This is also not surprising as it was the selected feature for both NPCC and PCC defaults. However, most of the example area can be classified correctly with an average of four instances of this feature, leaving the remaining template positions for other features to improve performance. Boundary accuracy and edge term improve with the remaining functions, typically 90° and 135° orientations of the small scale Gabor standard deviation feature. These features make intuitive sense for edge performance in this image as they are aligned with the orientation of the road.

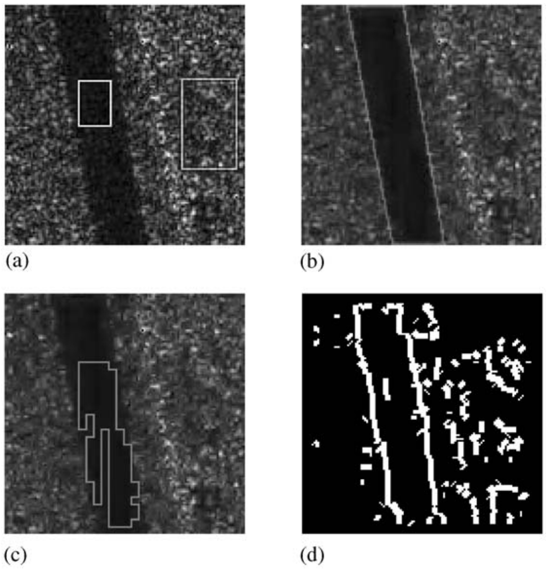
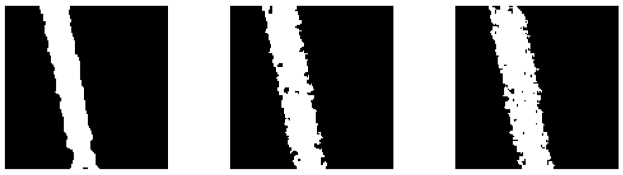


Fig. 6. Paved road vs. grass: (a) original image with examples (yellow/blue) and counter-examples (green/purple), (b) denoised image with ground truth (red), (c) region growing (red) and (d) edge results.

4.1.3. *Example 3: river vs. field*

The data and the results for river vs. field are shown in Figs. 9 and 10. The region growing did well, capturing almost all of the river class and none of the field class. The edge detection captures all of the rivers edges, with very few gaps (Fig. 8).

However, several false edges and texture edges are also detected, many of which will be included for fitness evaluation. Evolved results perform better than the defaults in terms of both the region term and the edge term. Demonstrating the effectiveness of the fitness function, example accuracy is higher for the evolved templates as well. Unfortunately, counter-example accuracy is significantly lower than that of the defaults. Despite the fact that the PCC of evolved templates is significantly lower due to counter-example performance, example accuracy improved enough that in terms of the NPCC evolved template performance is significantly better. Because the example class comprises only 17% of the image, the NPCC is the more applicable term. The intensity and 3×3 mean features perform most of the classification and thus use most of the positions in the template. On average



5	5	5
5	5	5
5	5	5

0	0	0
0	0	0
0	0	0

0	0	17
17	0	18
22	22	12

Evaluation Quantities	Bayes	NPCC/PCC	GA Result	Ave. GA Result
Region Term	0.861	0.953	0.970	0.963
Edge Term	0.099	0.513	0.599	0.554
Fitness	0.240	0.367	0.393	0.380
EA	0.672	0.809	0.846	0.827
CA	1.000	1.000	0.999	0.999
PCC	0.923	0.955	0.963	0.958
NPCC	0.836	0.904	0.923	0.914

Fig. 7. Paved road vs. grass results.

there are seven instances of some combination of these two features (approximately five of intensity and two of 3×3 mean). Thus, NPCC/PCC single feature defaults are incorporating too much smoothing and the genetic algorithm combines features in an effort to produce the correct amount of smoothing. Remaining positions are filled solely with Gabor standard deviation features, which help both for region and edge term performance. The most frequently selected of these are the small scale 0° and 135° orientations.

4.1.4. *Discussion on two-class experiments*

We have carried out these experiments on many images. In addition to the three two-class examples discussed above, we have also used lake vs. grass, lake vs. field, unpaved road vs. field and grass vs. field. We find that evolved templates consistently outperform the Bayesian default. Only in highly textured examples where fitness was misguided did the improvement over the Bayesian default fall below 1% for both PCC and NPCC. For all other experiments, the improvements were dramatically better. It is clear that example accuracy improved over the default for most examples. Furthermore, counter-example accuracy typically stabilized or improved due to post-processing.

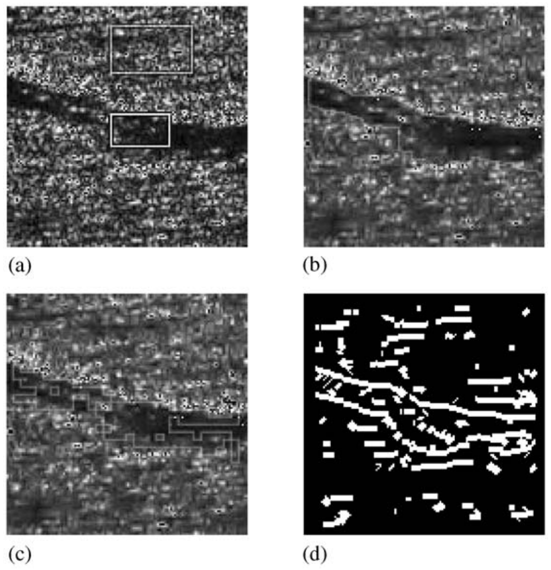


Fig. 8. River vs. field: (a) original image with examples (yellow/blue) and counter-examples (green/purple), (b) denoised image with ground truth (red), (c) region growing (Red) and (d) edge results.

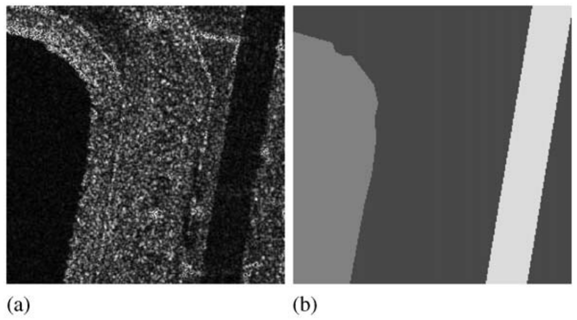
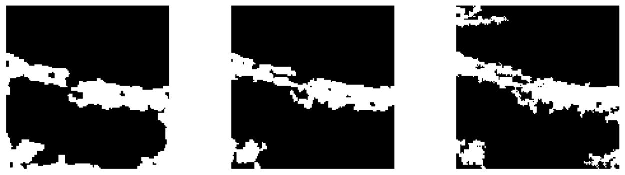


Fig. 10. (a) Original image and (b) ground truth image (lake-red, grass-blue, road-green).



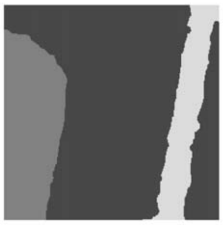
5	5	5
5	5	5
5	5	5

1	1	1
1	1	1
1	1	1

0	1	0
20	0	0
0	19	0

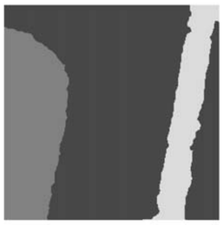
Evaluation Quantities	Bayes	NPCC/PCC	GA Result	Ave. GA Result
Region Term	0.726	0.708	0.764	0.739
Edge Term	0.269	0.323	0.400	0.363
Fitness	0.249	0.258	0.291	0.276
EA	0.706	0.687	0.758	0.730
CA	0.918	0.969	0.931	0.945
PCC	0.882	0.921	0.902	0.909
NPCC	0.812	0.828	0.844	0.838

Fig. 9. River vs. field results.



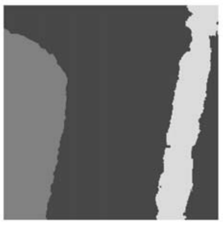
Bayesian Default

	Grass	Road	Lake
Grass	0.928	0	0.026
Road	0	0.840	0.038
Lake	0	0.003	0.999



NPCC/PCC Default

	Grass	Road	Lake
Grass	0.973	0	0.010
Road	0	0.868	0.031
Lake	0.001	0.003	0.999



GA Result

	Grass	Road	Lake
Grass	0.944	0	0.020
Road	0	0.898	0.024
Lake	0	0.023	0.994

	Bayes Default	NPCC/PCC Default	GA Result	Ave. GA Result
PCC	0.960	0.974	0.969	0.964
NPCC	0.922	0.947	0.946	0.939

Fig. 11. Lake vs. paved road vs. grass results.

The reasons for improvement over the Bayesian single feature default are:

- Examples and counter-examples may not fully characterize the classes.
- If a priori probabilities are assumed to be equal, the weighting of each pixel in terms of classification error is equal as in the PCC. This is biased for images with large discrepancies between example class and counter-example class.
- Placing the Bayesian best feature in all template positions generally implies an additional smoothing of the result.

In addition, evolved results perform at least as well or better than the NPCC/PCC defaults. Note, however, that while Bayesian and evolved results train only on the example and counter-example regions, NPCC/PCC defaults use the ground truth for the entire image to select the best feature. Despite this disadvantage, evolved results typically outperform NPCC/PCC defaults at example boundaries. Thus, for the set of experiments with higher example region perimeter/area ratio, (paved roads and the river examples) the evolved results are significantly better. We have found that when the perimeter/area ratio of the example region is lower, the evolved results are consistent with the NPCC/PCC defaults.

Most improvement due to evolution occurs within the first few generations. At this time the GA is typically optimizing the region term, until it reaches a plateau. However, the improvement in early generations can be misleading. This is improvement of the initial random templates of the population, not over default templates. Later generations typically refine the segmentation by optimizing the edge term. These optimizations may have less effect in the overall improvement from randomness but are crucial to improvements in boundary accuracy, which typically is the margin of improvement over the single feature default templates.

4.2. *N*-class image segmentation—lake vs. paved road vs. grass

Fig. 10 presents the original image and ground truth for this experiment. The original image is 256 × 256, and contains 22.5% Lake pixels (red), 14.7% Road pixels (green), and 62.8% Grass pixels (blue). *N*-class segmentation results are generated using hierarchical 2-class classifiers. The grass classifier is applied first, followed by the road classifier, and finally the lake classifier. Any pixel not assigned to one of these classes is handled by majority filtering. Fig. 12 shows the experimental results after false positive removal, false negative fill-in, and majority filtering. The single feature defaults and a typical GA result are given, along with confusion matrices and a table summarizing their NPCC/PCC performance. Although the evolved segmentation performs better than any default for the road region, it does not perform as well as the NPCC/PCC default for the grass and lake regions. Thus, evolved segmentations are significantly better than the Bayesian default and are consistent with the NPCC/PCC default (Fig. 11).

4.3. *Evaluation of functional template design*

The design of functional templates can be empirically evaluated by testing templates designed for a specific class on similar images. To accomplish this, some data must be preserved from the training of the template: the example set mean and standard deviation, discriminating functions, and Bayesian weights associated with discriminating functions.

4.3.1. *Example: lake vs. grass*

Fig. 12 shows the training and testing images used in this experiment. Both are 128 × 128 images containing visually similar data. Fig. 13 summarizes the training phase of the experiment showing the original image with example and counter-example set selection, denoised image with ground truth, as well as the region growing and edge detection results. To verify that the region growing result is correct, despite the missing areas in the middle of the lake region, the histogram equalized denoised image is also presented. It is visible from this image, Fig. 13(e), that the areas missed by region growing are statistically different. The final image in the figure is a typical evolved segmentation for the training image.

The results are presented in Table 2. The evolved templates (averaged over 10 runs with different random seeds) perform consistently with the NPCC/PCC single feature default segmentation for all criterion and outperform the Bayesian default segmentation by over 1% for all measures except counter-example accuracy, for which performance is consistent. These same evolved templates are applied to a second, testing image, to evaluate template design. The results on this testing image are slightly lower, but are still consistent with the training image for all measures. Furthermore, no single feature template outperforms the evolved result on the testing image. The average of the evolved results over 10 random seeds is consistent with the NPCC/PCC default for all criterion and significantly outperforms the Bayesian default.

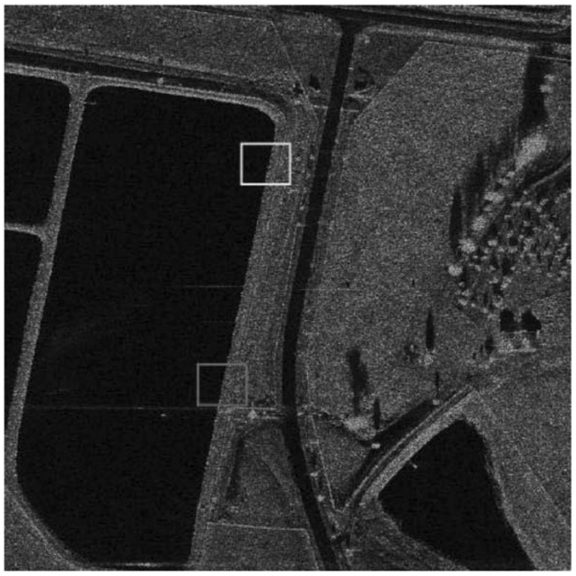


Fig. 12. Original SAR Image with training subimage (yellow) and testing subimage (red).

4.4. *Effectiveness of crossover operator*

Fig. 14 presents the genetic algorithm's fitness optimization, both with and without the crossover operator, for the first three examples and another example of lake vs. grass. After a small number of generations it is expected that a well designed crossover operator will outperform evolution with no crossover operator. For the system prototype, this trend is evident, within ten generations. The value of the fitness of the best individual of each generation is plotted for each system run. In all cases, the crossover operator improves the optimization performance of the genetic algorithm.

4.5. *Computational efficiency*

The system shown in Fig. 1 was implemented on a SUN Ultra II workstation which has a 200 MHZ CPU. To measure the system's capability for interactivity, several timing experiments were performed measuring efficiency of various segments of the prototype. Each of the user CPU time measurements presented in Table 3 is the average of measurements collected from four separate experiments on 128 × 128 images, each of which was run ten times. The total system user cpu time average is 40.96 s, given that the genetic algorithm is run for 10 generations. Of the total time the pre-learning portions of the code (not including post-processing)

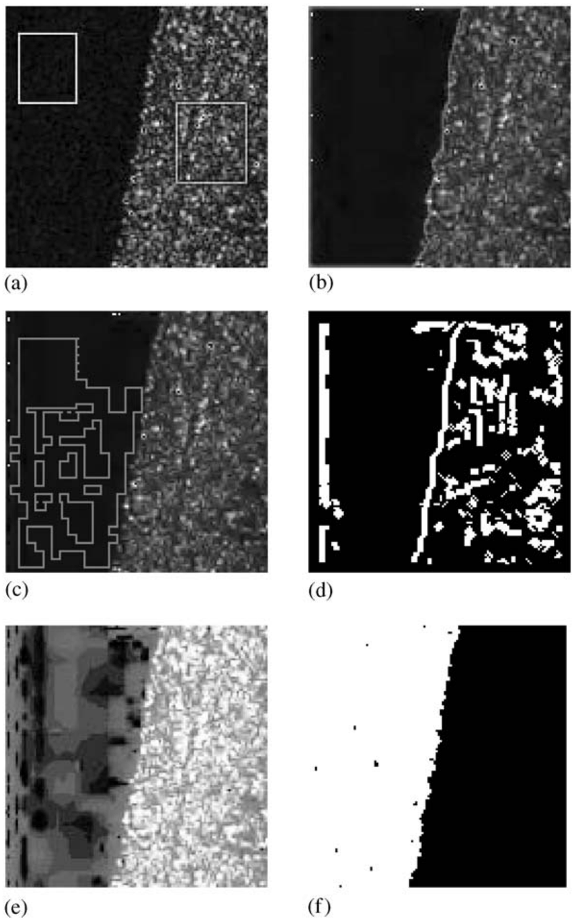


Fig. 13. Lake vs. grass training data: (a) original image with examples (yellow/blue) and counter-examples (green/purple), (b) denoised image with ground truth (red), (c) region growing (red), (d) edge results, (e) histogram equalization of denoised image and (f) GA segmentation result.

took 7.28 cpu seconds, approximately half of which is comprised of feature computation. However, for a completely interactive system, in which the user can review the segmentation result and refine example and counter-example selection, feature computation needs to be performed for the first pass only. Initialization of the genetic algorithm population took 5.2 cpu seconds, longer than other generations due to the need to generate and evaluate every individual in the population. Later generations, which perform selection, crossover, and mutation operations as well as evaluation of new individuals took on average 3.44 cpu seconds and represent the bottleneck of the approach, as would be expected. However, if the genetic algorithm is terminated after ten generations as in our experiments, evolution would take an average of 34 s, which is still within

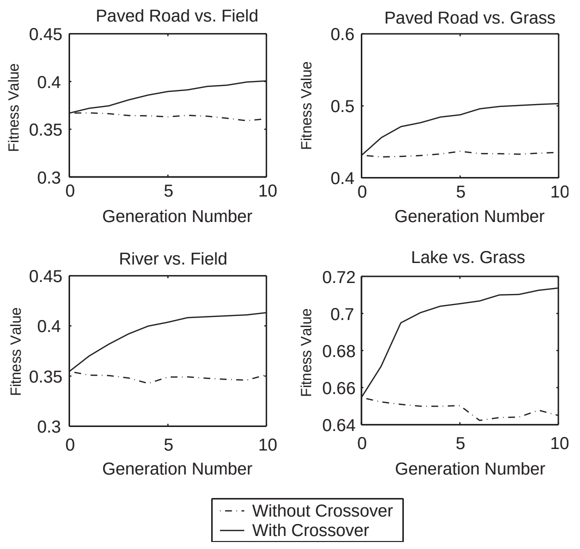


Fig. 14. Evolution of fitness both with and without crossover.

Table 2
Summary of lake vs. grass template design experiment

		Bayes default	NPCC/PCC default	GA result	Ave. (10 seeds)
	Template	1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0	0 19 17 0 0 20 0 0 0	
Training image 128 × 128	PCC	0.960	0.984	0.987	0.984
	NPCC	0.961	0.984	0.987	0.984
	EA	0.922	0.969	0.974	0.970
	CA	1.000	1.000	0.999	0.999
Testing image 128 × 128	PCC	0.951	0.979	0.980	0.979
	NPCC	0.962	0.984	0.985	0.983
	EA	0.924	0.967	0.970	0.967
	CA	1.000	1.000	1.000	1.000

reason for user interactivity. In addition, it should be noted that although efficiency was considered in the implementation of the prototype as noted in the approach, extreme attempts at optimizations were not made. Efficiency could be improved by (a) taking more extreme efforts in implementation, (b) using faster processors, and (c) utilizing parallelizable nature of the genetic algorithm. With such improvements not only the computation time can be significantly reduced but also larger populations and longer evolution periods could be considered while still maintaining interactivity, which would allow more of the search space to be explored.

5. Conclusions

Functional templates can be used for successful interactive segmentation and classification of SAR imagery using the approach developed in the system prototype discussed in this paper. Furthermore, functional templates can be retained for successful segmentation and classification of similar images in the future. The success of the prototype is described in four parts: genetic learning for functional template design, the physics-based segmentation evaluation, the crossover operator and the fitness function. Experimental results demonstrate that genetic learning is successfully applied for design of functional templates. Evolved templates select meaningful features, which complement each other for improved segmentation quality over any single feature. Evolved segmentations consistently outperform segmentations derived from the Bayesian best single feature and typically perform at least as well, if not better than segmentations derived from the actual best *single* feature defaults. Real SAR data was also used to illustrate the extension to N-class segmentation.

Physics-based segmentation incorporates SAR-specific information into the region growing and edge detection algorithms used for segmentation evaluation of the evolved templates. These algorithms, while not optimum, are used only in the example regions for comparisons to the evolved results and they have proven to be a suitable benchmark for optimizing the evolved results for the entire image. The crossover operator is successful at optimizing the fitness during evolution. This crossover operator provides a mechanism for preserving important spatial information about the arrangement of functions in the functional template. For all experiments, the fitness was clearly increasing as the functional templates evolved.

The margin of improvement for evolved segmentations typically occurs at the boundary of the image, illustrating the power of the fitness function. Given reasonable region growing and edge detection results the fitness function is effective for optimizing example accuracy. The region term successfully ensures correct classification of most of the pixels within a region (or at least the parts similar and adjacent to the example set). This term is typically improved to some level in the first few generations leaving the edge term for the focus of later generations. The edge term is most effective for perfecting results that are already good and for discouraging segmentations which classify the example region well but ‘overrun’ the region boundary.

Table 3
User CPU time measurements for code segments

Code	User CPU time (s)
Total computation time (10 generations)	40.96
Feature computation (FC)	3.57
Pre-learning computation time (incl. FC)	7.28
Generation 0	5.20
Ave/generation (after generation 0)	3.44

Acknowledgements

This work was supported in part by grant F49620-02-1-0315. The contents and information do not necessarily reflect the position or policy of US government.

References

[1] P. Andrey, Selectionist relaxation: genetic algorithms applied to image segmentation, Image Vision Comput. 17 (3–4) (1999) 175–187.

[2] B. Bhanu, X. Wu, Genetic algorithms for adaptive image segmentation, in: T. Poggio, S.K. Nayer (Eds.), Early Visual Learning, University Press, 1996.

[3] A.J. Katz, P.R. Thrift, Generating image filters for target recognition, IEEE Trans. Pattern Anal. Mach. Intell. 16 (9) (1994) 906–910.

[4] T. Sziranyi, M. Csapodi, Texture classification and segmentation by cellular neural network using genetic learning, Comput. Vision Image Understanding 71 (3) (1998) 255–270.

[5] B. Bhanu, S.K. Lee, Genetic Learning for Adaptive Image Segmentation, Kluwer Academic Publishers, Boston, 1994.

[6] T. Ferryman, B. Bhanu, A Bayesian approach for the segmentation of SAR images using dynamically selected neighborhoods, in: DARPA Image Understanding Workshop, Palm Springs, CA, USA, Vol. 2, 1996, pp. 1184–1186.

[7] S. Kulkarni, G. Lugosi, S. Venkatesh, Learning pattern classification—a survey, IEEE Trans. Inform. Theory 44 (8) (1998) 2178–2206.

[8] R. Delanoy, R. Lacoss, Analyst trainable fusion algorithms for surveillance applications, in: Proceedings SPIE Conference on Algorithms for Synthetic Aperture Radar Imagery V, Orlando, FL, USA, Vol. 3370, 1998, pp. 143–154.

[9] J.G. Verly, R.T. Lacoss, Automatic target recognition of LADAR imagery using functional templates derived from 3D CAD models, in: O. Firschein, T.M. Strat (Eds.), Reconnaissance, Surveillance, and Target Acquisition for the Unmanned Ground Vehicle: Providing Surveillance ‘Eyes’ for an Autonomous Vehicle, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1997 pp. 195–218.

[10] R. Chellappa, E. Zelnio, E. Rignot, Understanding synthetic aperture radar images, in: DARPA Image Understanding Workshop, San Diego, CA, USA 1992, pp. 229–247.

[11] A. Li, P. Dammert, G. Smith, J. Askne, Fuzzy c-means clustering algorithm for classification of sea ice and land cover from SAR images, in: Optical Engineering, Vol. 3217, 1997, pp. 86–97.
[12] A. Baraldi, F. Parmiggiani, Segmentation of SAR images by means of Gabor filters working at different spatial resolution, in: International Geoscience and Remote Sensing Symposium, Lincoln, NE, USA, Vol. 1, 1996, pp. 709–713.
[13] K. Chen, D. Tsay, W. Huang, Y. Tzeng, Remote sensing image segmentation using a Kalman filter-trained neural network, Int. J. Imaging Systems Technol. 7 (2) (1996) 141–148.
[14] E. Mingolla, W. Ross, S. Grossberg, A neural network for enhancing boundaries and surfaces in synthetic aperture radar images, Neural Networks 12 (3) (1999) 499–511.
[15] R. Sergi, G. Satalino, SIR-C polarimetric image segmentation by neural network, in: International Geoscience and Remote Sensing Symposium, Lincoln, NE, USA, Vol. 3, 1996, pp. 1562–1564.
[16] G. Guo, S. Ma, Unsupervised segmentation of SAR images, in: IEEE International Geoscience and Remote Sensing, Seattle, WA, USA, Vol. 2, July 1998, pp. 1150–1152.
[17] R. Schoenmakers, G. Wilkinson, T. Shouten, Results of a hybrid segmentation method, in: Proceedings SPIE Conference on Image and Signal Processing for Remote Sensing, Rome, Italy, Vol. 2315, September 1994, pp. 113–126.
[18] L. Gagnon, R. Klepko, Hierarchical classifier design for airborne SAR images of ships, in: Proceedings SPIE Conference on Automatic Target Recognition VIII, Orlando, FL, USA, Vol. 3371, April 1998, pp. 38–49.
[19] L. Soh, C. Tsatsoulis, Automated sea ice segmentation (ASIS), in: IEEE International Geoscience and Remote Sensing, Seattle, WA, USA, Vol. 2, July 1998, pp. 586–588.
[20] L. Soh, C. Tsatsoulis, Segmentation of satellite imagery of natural scenes using data mining, IEEE Trans. Geosci. Remote Sensing 37 (2) (1999) 1086–1099.
[21] S. Rogers, J. Colombi, C. Martin, J. Gainey, K. Fielding, T. Burns, D. Ruck, M. Kabrisky, M. Oxley, Neural networks for automatic target recognition, Neural Networks 8 (7–8) (1995) 1153–1184.
[22] H.J. Kim, D.W. Kim, S.K. Kim, J.U. Lee, J.K. Lee, Automatic recognition of car license plates using color image processing, Eng. Design Automation 3 (2) (1997) 217–225.
[23] M. Yu, N. Eau-anant, A. Saudagar, L. Udpa, Genetic algorithm approach to image segmentation using morphological operations, in: International Conference on Image Processing, Chicago, IL, USA, Vol. 3, October 1998, pp. 775–779.
[24] D.N. Chun, H.S. Yang, Robust image segmentation using genetic algorithm with a fuzzy measure, Pattern Recognition 29 (7) (1996) 1195–1211.
[25] S. Cagnoni, A. Dobrzeniecki, R. Poli, J. Yanch, Genetic algorithm-based interactive segmentation of 3D medical images, Image Vision Comput. 17 (12) (1999) 881–895.
[26] X. Yu, T. Bui, A. Krzyzak, The genetic algorithm parameter settings for robust estimation and range image segmentation and fitting, in: Proceedings of the Scandinavian Conference on Image Analysis, Tromso, Norway, Vol. 1, 1993, pp. 623–630.
[27] P. Zingaretti, A. Carbonaro, On increasing the objectiveness of segmentation results, in: International Conference on Advances in Pattern Recognition, Plymouth, UK, 1998, pp. 103–112.
[28] K. Delibasis, P. Undrill, G. Gameron, Designing texture filters with genetic algorithms: an application to medical images, Signal Process. 57 (1) (1997) 19–33.
[29] C. Jacquelin, G. Hejblum, A. Aurengo, Evolving descriptors for texture segmentation, Pattern Recognition 30 (7) (1997) 1069–1080.
[30] S. Masters, K. Hintz, Evolution of convolution kernels for feature extraction, in: Proceedings SPIE Conference on Signal Processing, Sensor Fusion, and Target Recognition IV, Orlando, FL, USA, Vol. 2484, 1995, pp. 536–544.
[31] K. Matsui, Y. Suganami, Y. Kosugi, Feature selection by genetic algorithm for MRI segmentation, Systems Comput. Jpn. 30 (7) (1999) 69–78.
[32] N. Campbell, B. Thomas, Automatic selection of Gabor filters for pixel classification, in: International Conference on Image Processing and its Applications, Dublin, Ireland, Vol. 2, 1997, pp. 761–765.
[33] A. Erdogan, K. Leblebicioglu, U. Halici, V. Atalay, Extraction of optimal texture features by a genetic algorithm, in: International Symposium on Computer and Information Sciences, Antalya, Turkey, Vol. 1, 1996, pp. 183–188.
[34] R.L. Delanoy, S.W. Troxel, Toolkit for image mining: user-trainable search tools, Lincoln Lab J. 8 (2) (1995) 145–160.
[35] R.L. Delanoy, J.G. Verly, D.E. Dudgeon, Machine intelligent automatic recognition of critical mobile targets in laser radar imagery, Lincoln Lab J. 6 (1) (1993) 161–212.
[36] J. Odegard, H. Guo, C. Burrus, R. Wells Jr., Wavelet based speckle reduction and image compression, Orlando, FL, USA in: SPIE, Vol. 2487, 1995, pp. 259–271.
[37] C. Oliver, S. Quegan, Understanding Synthetic Aperture Radar Images, Artech House Publishers, Norwood, MA, 1998.

About the Author—BIR BHANU received the S.M. and E.E. degrees in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology, Cambridge, the Ph.D. degree in Electrical Engineering from the Image Processing Institute, University of Southern California, Los Angeles, and the M.B.A. degree from the University of California, Irvine.

Currently Prof. Bhanu is the Director of the Center for Research in Intelligent Systems (CRIS) at the University of California, Riverside where he has been a Professor and Director of Visualization and Intelligent Systems Laboratory (VISLab) since 1991. Previously, he was a Senior Honeywell Fellow at Honeywell Inc. in Minneapolis, MN. He has been on the faculty of the Department of Computer Science at the University of Utah, Salt Lake City, UT, and has worked at Ford Aerospace and Communications Corporation, CA, INRIA-France and IBM San Jose Research Laboratory, CA. He has been the principal investigator of various programs for DARPA, NASA, NSF, AFOSR, ARO and other agencies and industries in the areas of learning and vision, image understanding, pattern recognition, target recognition, navigation, image databases, and machine vision applications. He is the co-author of books on *Computational Learning for Adaptive Computer Vision* (Forthcoming), *Genetic Learning for Adaptive Image Segmentation* (Kluwer, 1994), and *Qualitative Motion Understanding* (Kluwer, 1992). He has received two outstanding paper awards from the Pattern Recognition Society and has received industrial awards for technical excellence, outstanding contributions and team efforts. He has been the associate editor/guest editor of various journals and transactions. He holds 11 US and international patents and over 230 reviewed technical publications in the areas of his interest. He has been General Chair for IEEE Workshops on Applications of Computer Vision. Chair for the DARPA Image Understanding Workshop, General Chair for the IEEE Conference on Computer Vision and Pattern Recognition, Program Chair for the IEEE Workshops on Computer Vision Beyond the Visible Spectrum and Chair for IEEE Workshop on Learning in Computer Vision and Pattern Recognition.

Prof. Bhanu is a Fellow of IAPR, IEEE, AAAS and SPIE, and a member of ACM and AAAI.

About the Author—STEPHANIE FONDER graduated with a B.A. in Computer Science from the University of Minnesota, Morris in 1996. While working on her B.A., she also worked on automated target recognition research at the University of California, Riverside. She then continued on to complete her M.S. in Computer Science at the University of California, Riverside in 1999, with a thesis in image segmentation and pattern recognition. Since completion of her education, she is a software engineer developing signal processing and automation for SONAR systems at Digital System Resources.