Learning Feature Agents for Extracting Terrain Regions in Remotely Sensed Images

Bir Bhanu and Yingqiang Lin
Center for Research in Intelligent Systems
University of California, Riverside, CA, 92521, USA
{bhanu, yqlin}@cris.ucr.edu

Abstract—**In this paper, we apply genetic programming (GP) with smart crossover and smart mutation to discover integrated feature agents that are evolved from combinations of primitive image processing operations to extract regions-of-interest (ROIs) in remotely sensed images. The motivation for using genetic programming is to overcome the limitations of human experts, since GP attempts many unconventional ways of combination, in some cases, these unconventional combinations yield exceptionally good results. Smart crossover and smart mutation identify and keep the effective components of integrated operators called "agents" and significantly improve the efficiency of GP. Our experimental results show that compared to normal GP, our GP algorithm with smart crossover and smart mutation can find good agents more quickly during training to effectively extract the regions-of-interest and the learned agents can be applied to extract ROIs in other similar images.**

Index Terms—**Feature synthesis, Genetic programming, ROI extraction, Smart crossover and smart mutation.**

I. INTRODUCTION

Image segmentation and labeling of terrain region is an important task in remote sensing application [8, 9]. The quality of this task is primarily dependent on the data and features used as input. There are many kinds of features that can be used, the question is what are the appropriate features or how to synthesize features, particularly useful for segmentation/ labeling, from the primitive features extracted from images. Usually, there are almost infinite ways of combining primitive features to form synthesized composite ones. It is the human image experts who, relying on their rich experience, figure out a smart way of combination. The task of finding a good combination is equivalent to finding a good point in the search space of *agents* formed by the combination of primitive operations (also called primitive operators) on images.

However, limited by their speed, previous experience or even bias, the human experts can only try a very limited number of conventional combinations. Genetic programming (GP), on the other hand, may try many unconventional ways of combination that may yield exceptionally good results. Also, genetic programming can explore a much larger portion of agent space due to the inherent parallelism of GP and the speed of computer. The search performed by GP is guided by the goodness of agents in the population. As the search proceeds, GP will gradually shift the population to the portion of the space that contains good agents. The crossover operation is the major mechanism used by GP to search the agent space and the mutation operation is employed to increase the diversity of the population to avoid the premature convergence. However, in the traditional crossover and mutation, their locations are randomly selected, leading to disrupting the effective components (subtree in this paper) within agents and thus greatly reducing the efficiency of GP. It is very important for GP to identify and keep those effective components.

In this paper, we use genetic programming to generate feature agents for terrain labeling. The individuals in our GP based learning are agents 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 generated from the original image. The primitive feature images are pre-determined, and they are not the output of the pre-specified primitive operators. After applying an agent on the original image or the primitive feature images, the output image of the agent is segmented to yield a binary image or mask. The binary mask is used to extract the region containing the object from the original image. To improve the efficiency of GP, we propose *smart crossover* and *smart mutation* to identify and keep the effective components of an agent. The identification is performed on the basis of analyzing the interactions among the nodes of an agent. It is worth noting that the primitive operators and primitive features in our system are very basic and domain-independent, not specific to a kind of imagery, so our system can be applied to a wide variety of images.

II. RELATED RESEARCH

Genetic programming, an extension of genetic algorithm, was first proposed by Koza in [1] and it has been used in image processing, object detection, object recognition, etc. Poli [2] 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 [3] used GP to generate agents for ROI extraction. Their experimental results show that GP is a viable way to search the agent space. They also found that random selection of crossover and mutation points may degrade the performance of GP. Stanhope and Daida [4] used GP paradigms for the generation of 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. [5] applied GP to automatic detection of ships in low-resolution SAR imagery using an approach that evolves detectors. Roberts and Howard [6] used GP to develop automatic object detectors in infrared images. To improve the efficiency of GP, Ito et al. [7] proposed a depth-dependent crossover for GP in which the depth selection ratio is varied according to the depth of a node. A node closer to the root node of the tree has a better chance of being selected as a crossover point. Their experimental results show the superiority of the depth-dependent crossover to the random crossover in which crossover points are randomly selected.

Unlike the work of Stanhope and Daida [3] and Howard et al. [4], the input and output of each node of the tree in our system are images, not real numbers. Also unlike the work of Ito [7] that used only the syntax of the tree (the depth of a node), the smart crossover and smart mutation proposed in this paper evaluate the performance at each node to determine the interactions among them and use the fitness value at each node to determine the crossover and mutation points.

1. ADD_OP: A + B. Add two images.
2. SUB_OP: A – B. Subtract image B from image A.
3. MUL_OP: A * B. Multiply images A and B.
4. DIV_OP: 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_OP: A max B: Each pixel in the resultant image takes the larger value of the corresponding pixels in images A and B.
6. MIN2_OP: A min B. Each pixel in the resultant image takes the smaller value of the corresponding pixels in images A and B.
7. ADD_CONST_OP: A + c. Increase pixel value by c.
8. SUB_CONST_OP: A - c. Decrease pixel value by c.
9. MUL_CONST_OP: A * c. Multiply pixel value by c.
10. DIV_CONST_OP: A / c. Divide pixel value by c.
11. SQRT_OP: sqrt(A). For each pixel p with value v, if $v \geq 0$, change its value to $\sqrt{v}$. Otherwise, to $-\sqrt{-v}$.
12. LOG_OP: log(A). For each pixel p with value v. if $v \geq 0$, change its value to log(v). Otherwise, to –log(-v).
13. MAX_OP: max(A). Replace the pixel value by the maximum pixel value in a 3×3, 5×5 or 7×7 neighborhood.
14. MIN_OP: min(A). Replace the pixel value by the minimum pixel value in a 3×3, 5×5 or 7×7 neighborhood.
15. MED_OP: med(A). Replace the pixel value by the median pixel value in a 3×3, 5×5 or 7×7 neighborhood.
16. REVERSE_OP: rev(A). Reverse the pixel value. Suppose the maximum and minimum pixel values of image A are *Vmax* and *Vmin* respectively. If a pixel has value v, change its value to $Vmax - v + Vmin$.
17. STDV_OP: stdv(A). Obtain standard deviation image of image A by applying a template of size 3×3, 5×5 or 7×7.

Figure 1. Seventeen primitive operators (A and B are images of the same size and c is a constant).

III. TECHNICAL APPROACH

In our GP based approach, individuals are agents represented by binary trees. The search space of GP is the space of all possible feature agents. The space is very large. To illustrate this, consider only a special kind of binary tree, where each tree has exactly 30 internal nodes and one leaf node and each internal node has only one child. For 17 primitive operators (used in this paper) and only one primitive feature image, the total number of such trees is 17^{30}. It is extremely difficult to find good agents 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 run, and the criterion for terminating a run.

1) *The set of terminals* - the set of terminals used in this paper are seven primitive feature images generated from the original image: the first one is the original image; the others are mean and standard deviation images obtained by applying templates of sizes 3×3, 5×5 and 7×7. These images are the input to the agents. GP determines which operations are applied on them and how to combine the results.

2) *The set of primitive operators* - a primitive operator takes one or two images as input image(s), performs some primitive operations on them and stores the result in a resultant image. Currently, 17 primitive operators are used by GP to compose agents and they are shown in Figure 1.

3) *The fitness measure* - the fitness value of an agent is computed in the following way. Suppose during training G and G' are foregrounds in the ground truth image and the resultant image of the agent 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.

4) *Major parameters controlling the run* - the key parameters are the population size M, the number of generations N, the crossover rate and the mutation rate.

5) *Termination* - The GP stops whenever it finishes the pre-specified number of generations or whenever the best agent in the population has fitness value greater than the fitness threshold.

3.2. Reproduction, Crossover and Mutation

The GP searches through the agent space to find good agents. By searching through the agent space, GP gradually adapts the population of agents from generation to generation and improves the overall fitness of the whole population. The search is done by reproduction, crossover and mutation operations. The initial population is randomly generated and the fitness of each agent is evaluated. The reproduction operation involves selecting an agent from the current population. In this research, we use tournament selection, where a number of agents are randomly selected from the current population and the one with the highest fitness value is copied into the new population.

1) *Crossover:* Traditionally, to perform crossover, two agents are selected on the basis of their fitness values. These two agents are called parents. One internal node in each of these two parents is randomly selected, and the two subtrees with these two nodes as root are exchanged between the parents. In this way, two new agents, called offspring, are created. The two internal nodes selected during crossover are called crossover point. Since crossover point is selected randomly, the traditional crossover is also called *random crossover*. Usually, an agent with high fitness value contains a component (subtree) that is efficient in extracting the region of interest. However, due to the random selection of crossover point, random crossover may select an internal node within this component and disrupt it, leading to the great reduction in the efficiency of genetic programming.

In order to avoid the above problem, we propose *smart crossover* that identifies and keeps the good components of an agent. In the smart crossover, the output image of each node, not just the resultant image from the root node, is evaluated and the fitness value of each node is recorded. Based on the fitness values of all the nodes, the edges of an agent are classified as good edge or bad edge. If the fitness value of a node is smaller than that of its parent node, the edge linking these two nodes is a good edge. Otherwise, the edge is labeled as bad edge. During crossover, all the bad edges are identified and one of them is randomly selected. The child node of the selected bad edge is chosen as the crossover point. If an agent has no bad edge, the crossover point is selected randomly.

We use ε-greedy policy to determine which crossover is used. With probability ε, the smart crossover is invoked; with probability 1 - ε, the random crossover is invoked. In this paper, ε is set to 0.9.

2) *Mutation:* In order to avoid premature convergence, mutation is introduced to randomly change the structure of some agents to help maintain the diversity of the population. The agent selected for mutation is selected randomly. Also, both *random mutation* and *smart mutation* are used. In the random mutation, a node of an agent is randomly selected as mutation point, and the subtree rooted at this node is replaced by another randomly generated tree. The resulting new agent replaces the old one in the population. In the smart mutation, one of the bad edges is randomly chosen, and the child node of the bad edge is selected as mutation point. If the parent node has only one child, the parent node is deleted and the child node is directly linked to the grand parent node (the parent node of the deleted parent node); if the parent node has two children, the parent node and the subtree rooted at the child node with smaller fitness value are deleted and the other child node is directly linked to the grand parent node. If the grand parent node does not exist, the child becomes the root node. The smart mutation attempts to delete internal node that decreases the fitness value and keeps the good component. If the agent has no bad edge, the mutation point is selected randomly. Similar to the crossover case, ε-greedy policy determines which mutation is invoked.

3.3. Steady-state and Generational Genetic Programming

In *steady-state GP*, two agents are selected based on their fitness values for crossover. The children of this crossover, perhaps mutated, replace a pair of agents with the smallest fitness values. The two children are executed immediately and their fitness values are recorded. Then another two agents are selected for crossover. This process is repeated until crossover rate is satisfied.

In *generational GP*, two agents are selected based on their fitness values for crossover and the two offspring of the crossover are kept. At this time, the two offspring are not put into the current population and they will not participate in the following crossover operations. The above process is repeated until crossover rate is satisfied. After crossover operations are finished, all the children resulting from the crossover operations are combined with the current population and the tournament selection is employed to select agents from the combined population to generate the new population of next generation.

For both GP algorithms, we adopt an *elitism* replacement method to copy the best agent from generation to generation.

IV. EXPERIMENTS

Various experiments were performed to test the efficacy of genetic programming in extracting regions of interest from real synthetic aperture radar (SAR) images. In this paper, we show three selected examples. It is worth noting that the training and testing images are different and the ground truth is used only during the training. In all the experiments, the maximum size of agent is 30 and the threshold value for segmentation is 0. In each experiment, both normal genetic programming (GP with random crossover and random mutation) and smart genetic programming (GP with smart crossover and smart mutation) are applied to the training image. For the purpose of objective comparison, we invoke the normal GP and smart GP with the same set of parameters in each experiment. The population size is 100, the number of generations is 100, the fitness threshold value is 0.90 in the

first two experiments and 0.95 in the third experiment, the crossover rate is 0.6, and the mutation rate is 0.1. The experimental results are shown in Table 1, where the "Best fitness" column and the "Average fitness" column show the fitness of the best agent and the population fitness (the average fitness of all the agents in the population) after a particular generation. The numbers in the parenthesis in the "Best fitness" columns are the fitness values of the best agents on the testing SAR images. A comparison of the performance between normal GP and smart GP is shown in Figures 2 and 3. Figure 2 shows how the fitness of the best agent improves as the normal GP and smart GP run. Similarly, Figure 3 shows the improvement of the population fitness as normal GP and smart GP run. Both Figures 2 and 3 show that smart GP finds good agents more quickly. Since population fitness is the average fitness values of 100 agents in the population, Figure 3 demonstrates convincingly that the smart GP is more efficient.

Table 1. The performance of normal GP and smart GP on various SAR images during training (and testing).

Normal genetic programming								
• River vs. Field			• Field vs. Grass			• Lake vs. Field		
Generation	Best fitness	Average fitness	Generation	Best fitness	Average fitness	Generation	Best fitness	Average fitness
0	0.43	0.21	0	0.62	0.44	0	0.65	0.42
48	0.74 (0.84)	0.68	50	0.87 (0.68)	0.86	45	0.93 (0.92)	0.92
Smart genetic programming								
• River vs. Field			• Field vs. Grass			• Lake vs. Field		
Generation	Best fitness	Average fitness	Generation	Best fitness	Average fitness	Generation	Best fitness	Average fitness
0	0.31	0.14	0	0.62	0.47	0	0.69	0.49
3	0.56	0.25	5	0.85	0.60	5	0.91	0.85
19	0.75	0.69	32	0.89	0.86	17	0.93	0.89
52	0.79 (0.81)	0.75	94	0.90 (0.83)	0.89	26	0.94	0.92

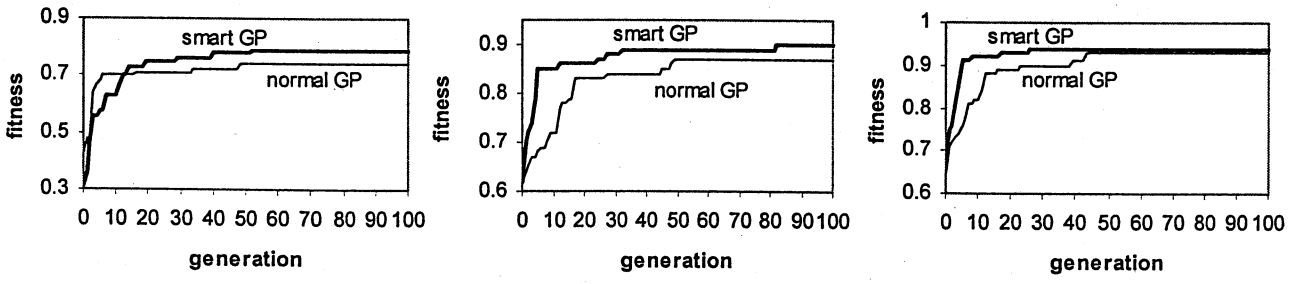


Figure 2. The fitness of the best agent versus generation.

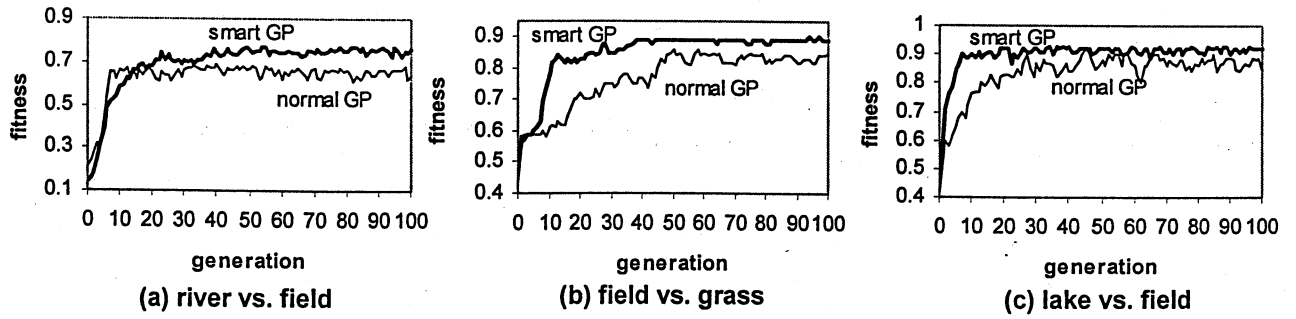


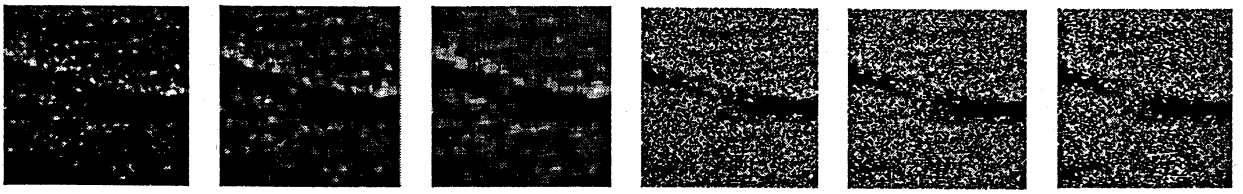
Figure 3. The fitness of agent population versus generation.

1) *Extracting river from real SAR images:* We have two SAR images containing river shown in Figure 4. Figure 4 (a) was used in training and GP generated an agent to extract the river in the image. The first primitive feature image is the original image and the mean and standard deviation images are shown in Figure 5, where 3×3, 5×5 and 7×7 are the size of the template used to obtained this primitive feature image from the original image. The generated agent was applied to the SAR image shown in Figure 4 (d) to test its efficacy in extracting the river. The steady-state genetic programming

was used to generate the agent. Figure 4 (b) shows the regions of interest extracted by the best agent generated by normal GP. The fitness value of the best agent is not very good (see Table 1). Two reasons account for this. First, the river in Figure 4 (a) accounts for a small percentage of the total area in the image. Second, there are some islands in the river. These islands are similar to the field, but they are not excluded from the ground truth. Figure 4 (c) shows the regions of interest extracted by the best agent generated by smart GP, and the best agent is shown in Figure 6, where PM_IM0 is original image, PF_IM1 is 3×3 mean image and PF_IM4 is 5×5 standard deviation image. It has 30 internal nodes and 17 of them contain MED_OP primitive operator, which is very useful in speckle noise reduction. It is possible to have a more compact tree representation of this agent. It can be seen from Table 1 that smart GP is more efficient. We applied the two best agents to the image shown in Figure 4 (d). Figure 4 (e) shows the region of interest extracted by the best agent from normal GP. The fitness value of the agent is 0.84. This number is larger than the fitness value in the training because the river in Figure 4 (d) accounts for a much larger percentage of the total area of the image than in Figure 4 (a). Figure 4 (f) shows the region of interest extracted by the best agent from smart GP. The fitness value is 0.81.

(a) river vs. field (training) (b) best ROI in training (normal GP) (c) best ROI in training (smart GP) (d) river vs. field (testing) (e) best ROI in testing (normal GP) (f) best ROI in testing (smart GP)

Figure 4. SAR images containing river and field.

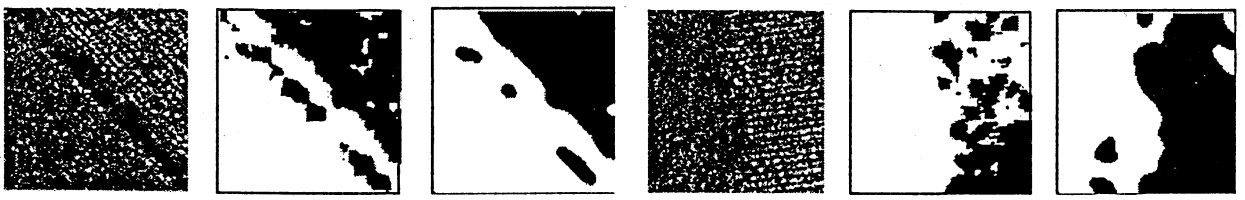


(a) 3×3 mean image (b) 5×5 mean image (c) 7×7 mean image (d) 3×3 stdv image (e) 5×5 stdv image (f) 7×7 stdv image

Figure 5. Primitive feature images of training SAR image containing river and field.

```
(LOG_OP (MUL_CONST_OP (MED_OP (MED_OP (MED_OP (MED_OP (MED_OP (MED_OP (MED_OP
(MED_OP (MED_OP (MED_OP (MED_OP (MED_OP (MED_OP (MED_OP (MED_OP (MED_OP (MED_OP
(DIV_CONST_OP (ADD_CONST_OP (MIN2_OP (SUB_OP (ADD_CONST_OP (MIN_OP (SUB_CONST_OP
PM_IM1))) (STDV_OP (STDV_OP (SQRT_OP (MAX2_OP PF_IM0 PF_IM0))))) PF_IM4))))))))))))))))))))
```

Figure 6. Learned agent in LISP notation by smart GP.



(a) field vs. grass (training) (b) best ROI in training (normal GP) (c) best ROI in training (smart GP) (d) field vs. grass (testing) (e) best ROI in testing (normal GP) (f) best ROI in testing (smart GP)

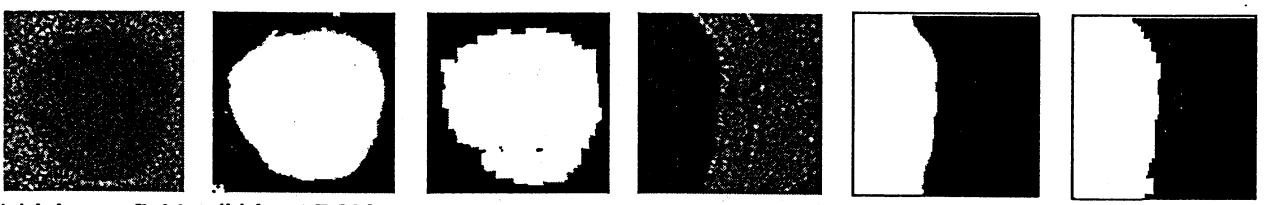
Figure 7. SAR images containing field and grass.

2) *Extracting field from real SAR images:* Figure 7 shows two SAR images containing field and grass. Figure 7 (a) was used in training and Figure 7 (d) was used in testing. We consider extracting field from a SAR image containing field and grass as the most difficult task among the three experiments, since the grass and field are similar to each other.

The generational genetic programming was used to generate the agent. Figure 7 (b) shows the region extracted by the best agent found by normal GP and Figure 7 (c) shows the region extracted by the best agent found by smart GP. Table 1 shows that smart GP searches the agent space more efficiently than normal GP. We applied the two best agents to the image in Figure 7 (d). Figure 7 (e) shows the region of interest extracted by the best agent from normal GP. The fitness value of the agent is 0.68. Figure 7 (f) shows the region of interest extracted by the best agent from smart GP. The fitness value of the agent is 0.83. The best agent evolved by smart GP is much better than that evolved by normal GP.

3) *Extracting lake from real SAR images:* Two SAR images contain lake. The first one contains a lake and field, and the second one contains a lake and grass. Figure 8 (a) shows the original image containing lake and field and Figure 8 (d) shows the image containing lake and grass. We used the SAR image containing the lake and field as the training image and applied the agent generated by GP to the SAR image containing the lake and grass. The steady-state genetic programming was used to generate the agent. Figure 8 (b) shows the region extracted by the best agent from normal GP and Figure 8 (c) shows the region extracted by the best agent from smart GP. Smart GP finds the good agents more quickly than normal GP. We applied the two best agents to the image in Figure 8 (d). Figure 8 (e) shows the region of interest extracted by the best agent from normal GP. The fitness value of the agent is 0.92. Figure 8 (f) shows the region of interest extracted by the best agent from smart GP. The fitness value of the agent is 0.97, which is higher than that from normal GP.

It is worth noting that since elitism mechanism is adopted, the fitness value of the best agent is not decreased from generation to generation. However, the average fitness value of the population may be decreased, although the fluctuation is very small. The general trend is that the average fitness values increases quickly within the first 20 generations. Then the marginal improvement can be achieved in the rest of the generations.



(a) lake vs. field (training) (b) best ROI in training (normal GP) (c) best ROI in training (smart GP) (d) lake vs. grass (testing) (e) best ROI in testing (normal GP) (f) best ROI in testing (smart GP)

Figure 8. SAR image containing lake and field.

V. CONCLUSIONS

In this paper, we presented a basic approach that uses genetic programming to evolve agents for extracting terrain regions in remotely sensed images. In order to improve the efficiency of genetic programming, we proposed smart crossover and smart mutation that can identify and keep the effective ROI extraction components in agents. We used SAR imagery to demonstrate the approach for extracting terrain regions. Our experimental results showed that GP can find good agents to effectively extract ROIs in SAR images and the smart crossover and smart mutation make GP find these good agents more quickly, thus greatly improving the efficiency of GP. The agents generated by GP during the training can be applied to extract ROIs in other similar images. Currently, in order to get the fitness at each node, we have to evaluate its output image, which is a time consuming and inefficient process. To further improve the efficiency of GP, it is important to find a way to estimate the fitness of internal nodes based on the fitness of the root node.

Acknowledgment: This research is supported by AFRL grant F33615-99-C-1440 and NSF grant IIS-0114036. The contents of the information do not necessarily reflect the position or policy of the U. S. Government

REFERENCES

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

[2] R. Poli, "Genetic programming for feature detection and image segmentation," in *Evolutionary Comp.*, T. C. Forgarty Ed., pp. 110-125, 1996.

[3] B. Bhanu and Y. Lin, "Learning composite operators for object detection", *Proc. Genetic and Evolutionary Computation Conference*, July, 2002.

[4] S. A. Stanhope and J. M. Daida, "Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery," *Proc. Conf. Evolutionary Programming VII*, pp. 735-744, 1998.

[5] D. Howard, S. C. Roberts, and R. Brankin, "Target detection in SAR imagery by genetic programming," *Advances in Engineering Software*, 30(5), pp. 303-311, May 1999.

[6] S. C. Roberts and D. Howard, "Evolution of vehicle detectors for infrared line scan imagery," *Proc. Workshop, Evolutionary Image Analysis, Signal Processing and Telecommunications*, pp. 110-125, 1999.

[7] T. Ito, H. Iba and S. Sato, "Depth-dependent crossover for genetic programming," *Proc. IEEE International Conference on Evolutionary Computation*, New York, NY, USA, pp. 775-780, 1998.

[8] J. Mvogo, V. Onana, J. Rudant, H. Trebossen, G. Mercier and E. Tonye, "Coastline detection in SAR images using wavelet packets, multiscale segmentation and a Markov random field regularization," *Proc. SPIE - The International Society for Optical Engineering*, vol. 4173, pp. 122-131, 2000.

[9] Y. Dong, A. Milne and B. Forster, "Segmentation and classification of vegetated areas using polarimetric SAR image data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, (no.2), pp. 321-329, Feb. 2001.

Proceedings of the 2nd PATTERN RECOGNITION FOR REMOTE SENSING WORKSHOP

PRRS 2002

Fallsview Sheraton Hotel, Niagara Falls, Canada

August 16, 2002

Co-sponsored by IAPR (TC7) and IEEE

Editor: Maria Petrou

BMVA Press, ISBN 1 901725 18 9