Recognition of Objects by Using Genetic Programming

Nerses Safaryan
Algorithmic Languages and Programming,
State Engineering University of Armenia
Yerevan, Armenia

Hakob Sarukhanyan
Digital signal and Image processing laboratory,
Institute for Informatics and Automation Problems of
National Academy of Sciences
Yerevan, Armenia

***Abstract*—This document is devoted to the task of object detection and recognition in digital images by using genetic programming. The goal was to improve and simplify existing approaches. The detection and recognition are achieved by means of extracting the features. A genetic program is used to extract and classify features of objects. Simple features and primitive operators are processed in genetic programming operations. We are trying to detect and to recognize objects in SAR images. Due to the new approach described in this article, five and seven types of objects were recognized with good recognition results.**

Keywords—Terminals; Fitness; Selection; Crossover; Mutation; Ground Truth

I. INTRODUCTION

Object recognition is still a challenge for computer vision systems in general. The main purpose of object recognition is to identify the kinds of the objects in an image [1]. Object recognition algorithms rely on matching or learning algorithms using appearance-based or feature-based techniques. We are trying to achieve good recognition results using the feature-based technique [2, 6]. The quality of object recognition is heavily dependent on the effectiveness of features. The features used to represent an object are the key to the object detection and recognition. It is difficult to extract good features from real images due to various factors, including noise. There are many features that can be extracted. It is very difficult to find appropriate features and to synthesize composite features. Synthesizing effective new features from primitive features is equivalent to finding good points in the feature combination space where each point represents a combination of primitive features. The feature combination space and feature subset space are huge and complicated and it is very difficult to find good points in such vast spaces unless one has an efficient search algorithm [5, 7]. Genetic programming (GP) is used as search algorithm. GP may try many unconventional combinations and in some cases these unconventional combinations yield exceptionally good recognition performance. Also, the inherent parallelism of GP and the speed of computers allows a much larger portion of the search space to be explored. We have used a simple (steady-state) [2] genetic programming algorithm and primitive features to detect and to recognize objects in SAR images. A program system is developed based on this GP algorithm by means of which two experiments were done: trying to recognize five types of objects, trying to recognize seven types of objects.

II. GENETIC PROGRAMMING IN OBJECT RECOGNATION

Individuals of GP are composite operators in task object recognition. The composite operators are represented by binary trees. Internal nodes of binary trees are primitive operators and leaf nods are primitive features [4]. GP uses five major considerations in the task of object detection and recognition:

A. The set of terminals:

The set of terminals include the following images: F_0, F_1, …, F_{15}, where F_0 is the original image, the F_1, F_2 and F_3 are 3x3, 5x5 and 7x7 mean images, the F_4, F_5 and F_6 are 3x3, 5x5 and 7x7 deviation images, the F_7, F_8 and F_9 are 3x3, 5x5 and 7x7 maximum images, the F_{10}, F_{11} and F_{12} are 3x3, 5x5 and 7x7 minimum images and the F_{13}, F_{14} and F_{15} are 3x3, 5x5 and 7x7 median images [4].

B. Primitive operators:

The primitive operators are given below in the Table 1 [2, 4].

TABLE I. THE SET OF TERMINALS

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

16	*MEAN*(*A*)	Replace the pixel value by the average pixel value of a 3×3, 5×5 or 7×7 neighborhood.
17	*STDV*(*A*)	Replace the pixel value by the standard deviation of pixels in a 3×3, 5×5 or 7×7 neighborhood.

C. Fitness value

The fitness value of a composite operator is computed in the following way. Suppose *G* and *G′* are foregrounds in the ground truth image and the resultant image of the composite operator respectively. Let n(X) denote the number of pixels within region X, then Fitness = n (G∩G′) / n (G∪G′) [2, 4].

D. Parameters and termination

We have used the following parameters for GP: Population size - M, the number of generation - N, the crossover rate, the mutation rate and the fitness threshold [3, 10]. The GP stops whenever it finishes the pre-specified number of generations or whenever the best composite operator in the population has fitness value greater than the fitness threshold.

E. Operations of Genetic programming

The search is done by performing selection, crossover and mutation operations in GP [2, 3].

a) Selection: genetic operators in GP are applied to individuals that are probabilistically selected based on fitness. Better individuals are more likely to have more child programs than inferior individuals. We are using tournament selection [3]. In tournament selection a number of individuals are chosen at random. These are compared with each other and the best of them is chosen to be the parent.

b) Crossover: we are using sub tree crossover [3]. Two composite operators are selected on the basis of their fitness values as parents: sub tree crossover randomly selects a crossover point in each parent tree, two subtrees rooted at these two nodes are exchanged between the parents to generate two new composite operators which are called offspring.

c) Mutation: mutation is introduced to randomly change the structure of some individuals. We are using tree type of mutation [2, 3]:

- Randomly select a node of the binary tree representing a composite operator and replace the sub tree rooted at this node, including the node selected, by another randomly generated binary tree.
- Randomly select a node of the binary tree representing a composite operator and replace the primitive operator stored in the node with another primitive operator of the same number of inputs as the replaced one. The replacing primitive operator is selected at random from all the primitive operators with the same number of input as the replaced one.
- Randomly select two sub trees within a composite operator and swap them. Of course, neither of the two sub-trees can be a sub-tree of the other.

III. Used algorithm of Genetic Progrmming

Steady-state genetic programming [2, 4] is used to synthesize composite operators. The first step of the algorithm is to randomly generate the initial population. We will use half-and-half method for generating the initial population [3, 4, 6]. Suppose there are a set of primitive operators O = {O_1, O_2, ..., O_N} and a set of terminals F={F_1, F_2, ..., F_M}. If we randomly generate population P of size M from O and F, we obtain their combinations, which consist of primitive operators and terminals, as members (composite operator) of P. Then each composite operator in P is evaluated and crossover operation is perform. If we show composite operators as a binary tree, they will be like on figure 1 A. We can have two offspring (composite operators) after crossover operation, which are shown in figure 1 B. These two offspring replace two of the worst composite operators in P. This is continued until crossover rate is met. Then mutation is performed, which can replace the primitive operator stored in the node with another primitive operator. For example, if we observe the tree of figure 1C, after mutation we can have the tree of figure 1D.

Finally we get the best individual (composite operator) by applying selection operation.

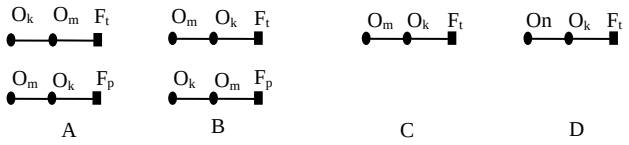

Fig. 1. GP operations are shown graphiclly

The steps of Steady-state genetic programming are folowing:

1) For gen = 1 to N do // N is the number of generation.
2) Keep the best composite operator in P.
3) Repeat.
4) Select 2 composite operators from P based on their fitness values for crossover.
5) Select 2 composite operators with the lowest fitness values in P for replacement.
6) Perform crossover operation and let the 2 offspring replace the 2 composite operators selected for replacement.
7) Execute the 2 offspring and evaluate their fitness values.
8) Until crossover rate is met.
9) Perform mutation on each composite operator with probability of mutation rate and evaluate mutated composite operators.
10) After crossover and mutation, a new population P' is generated
11) Let the best composite operator from population P replace the worst composite operator in P' and let P = P'.
12) if the fitness value of the best composite operator in P is above fitness threshold value then stop

IV. Program System Parts

The software system is developed using steady-state genetic programming. The software system consists of two phases: training (see figure 2) and testing (see figure 3).

Training image is an input image that includes an object of a type. For this type a class should be obtained: a composite operator and a template image. The block "Feature extractor" is a collection of operations for getting the set of terminals. Steady-state genetic programming is used in the block "Genetic programming". "Ground Truth" image is created manually from the input image.

Training part: The composite operator is synthesized for a definite class of images in the training part. Then segmented template image is obtained from the training image by means of the composite operator. The segmented [8, 9] template image and the composite operator are saved as identifiers of class to which the training image belongs. As a result we will have one or more composite operator/operators and one or more template image/images for a definite image. For different classes class identifiers are saved separately.

Testing part: Our system recognizes an image for which the system has a composite operator and a template image in the testing part. Recognition is performed in the following way: the composite operators from all the classes are run on the testing image (separately and parallelly), afterwards the results are segmented as binary images. Each class will have its segmented binary images. Then will the fitness of the resulting images for all classes is calculated separately: in the fitness function as Ground Truth image will be the template image, which corresponds to the composite operator.

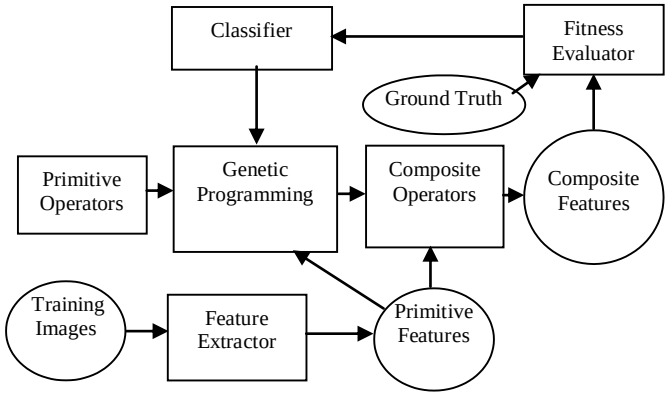

Fig. 2. Learning part diagram of programming system

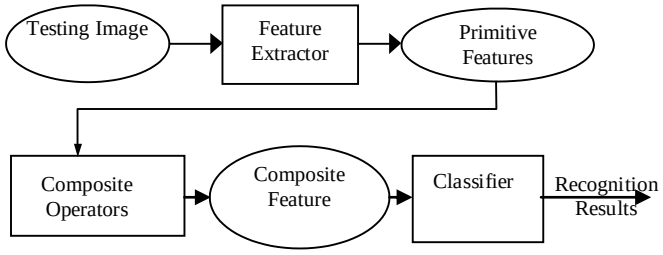

Fig. 3. Testing part diagram of programming system

V. Experiments

Two experiments are performed for recognizing different manmade objects in SAR images.

A. Experiment 1:

Composite operators and segmented binary template images for two kinds of objects (car and helicopter from A SAR image from MSTAR Image database) are found. A car from the SAR image (see figure 4a) is used to synthesize the composite operator, as Ground Truth image figure 4b is used.

Fig. 4. A car image region from real SAR image. a) Real image of car, b) Ground Truth, c) Binary segmented image after composite operator

Synthesized composite operator is flowing:

ADD(PFIM0,FMAX2(FDIV(ADD(FLOG(FDIVC(PFIM0)), FADDC(FDIV(FMUL(PFIM5, PFIM5), ADD(PFIM13, PFIM0)))), ADD(FSUBC(FADDC(PFIM10)), PFIM0)), FLOG(FDIV(PFIM10, FSUBC(FSUBC(FSUBC(PFIM0)))))))

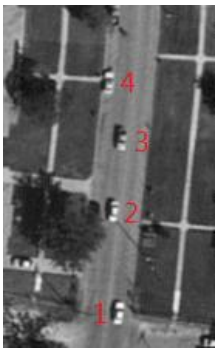

Fig. 5. Cut region from real SAR image

Fig. 6. Cut region from real SAR image

The result of composite operator is figure 4c. Figure 6 A is saved as class template to recognize cars like on figure 4a on the same azimuth angle. The composite operator was applied on car 1 in figure 5 to obtain class template for “car 1” like cars on the same azimuth angle, the result of which was figure 6 B image. Using the composite operator and 6 B class template the recognition rates of car 1, car 2 and car 4 were 0.95, 0.7 and 0.69 respectively. Car 3 was recognized as another object. The composite operator for recognition helicopter is the following:

ADD(PFIM14, ADD(FADDC(FMAX2(FSQRT(PFIM12), FMIN(FMIN(FDIV(PFIM12, PFIM12))))), FSUB(PFIM7, FMEAN(FMAX2(PFIM2, PFIM14))))), which is obtained by using figure 7 helicopter 1 as training image. As Ground Truth is used figure 8 A and the resulting image of the composite operator is shown in figure 8 B, which is saved as template

image for helicopter on the same azimuth angle. The composite operator and 8 B template image are used to recognize helicopters from fig. 7.1, 7.2, and 7.3. The results of recognition were 0.73, 0.7 and 0.73 respectively.

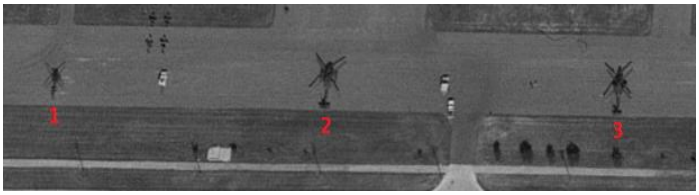

Fig. 7. Cut region from real SAR image

A B

Fig. 8. Cut region from real SAR image

B. *Experiment 2:*

The experiment was done on five types of object images from [2].

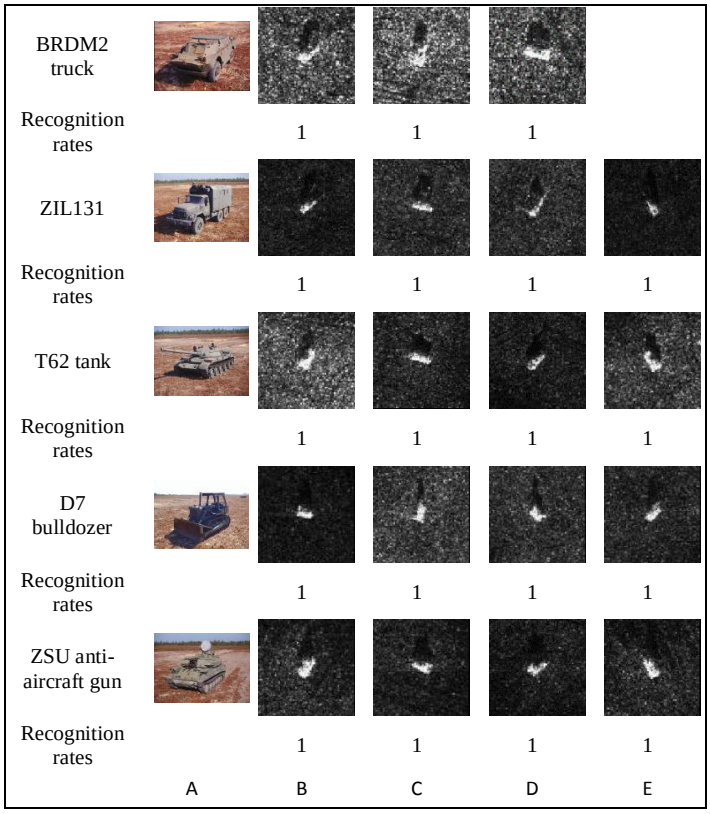

Fig. 9. SAR images of five types of objects and its recognition rates

Figure 9 shows one optical and four SAR images of each object under 15°-depression angle and various azimuth angles between 0° and 359°. One of the four images for each object are used as training image in our program system to get class identifiers for these five types of objects. Composite operators are synthesized for each class (type of object) separately. Class template images are obtained for each SAR images in figure 9 B to E, using those composite operators. As we already mentioned, our system is able to recognize the same object in range $-2^0 <= $ template azimuth angle$<=2^0$ by means of a class template image. The recognition rates of SAR images from figure 8 are equal to 1. Various experiments results are given in figure 10. As test images for our program system are given images obtained from SAR images from figure 8, rotated -2^0 to 2^0 angels. We achieve the same results in two cases (results are given in figure 9):

1) When our system recognizes these five objects.

2) When our system recognizes these five objects and added cars and helicopter objects.

The best experiment results from [2] are given in table 2. Experiments are done for three and five types of objects. We can see that results for five objects are worse than the results of three objects.

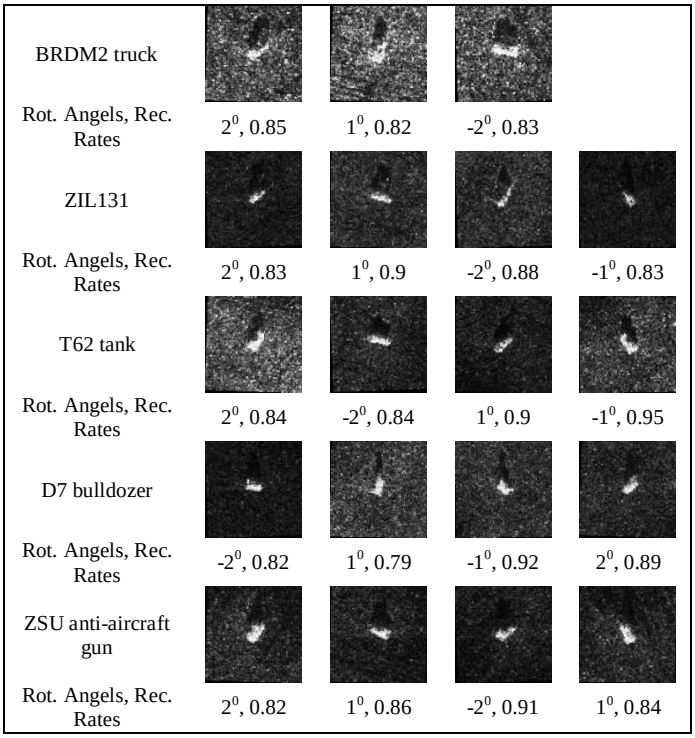

Fig. 10. Rotated SAR images of five types of objects and its recognition rates

TABLE II. EXPERIMENTS RESULTS FROM [2]

Experiment results for tree type objects			Experiment results for five type objects		
Runs	Training	Testing	Runs	Training	Testing
1	1.0	0.967	1	0.929	0.824
2	1.0	0.971	2	0.926	0.816
3	1.0	0.976	3	0.929	0.803
4	1.0	0.964	4	0.941	0.845
5	1.0	0.967	5	0.931	0.809
6	0.995	0.979	6	0.911	0.809
7	1.0	0.979	7	0.903	0.832
8	0.995	0.95	8	0.943	0.842
9	1.0	0.964	9	0.94	0.847

10	0.995	0.983	10	0.909	0.823
Average	0.999	0.97	Average	0.926	0.825

VI. CONCLUSION

A program system, which detects and recognizes objects in SAR images, was presented in this paper. A new approach was used for recognizing objects by using genetic programming to achieve the same good recognition rate, despite the increase of the quantity of object types. One of its benefits was the usage of the primitive set of terminals and the simple genetic programming algorithm.

REFERENCES

[1] http://en.wikipedia.org/wiki/Outline_of_object_recognition

[2] Bir Bhanu and Yingqiang Lin, "Object Detection in Multi-modal Images Using Genetic Programming," University of California, Riverside, CA, 92521, USA, October 2003.

[3] Riccardo Poli, William B. Langdon, Nicholas F. McPhee, John R. Koza, "A Field Guide to Genetic Programming," March 2008

[4] Nerses Safaryan, "Detection and Classification of Objects by Applying Genetic Programming", Mathematical Problems of Computer Science 32, pp. 101-106, 2009.

[5] Nerses Safaryan, "Generation Of Operators To Identificatify Of Objectss In Digital Images," (in russian), XXXVI Gagarin readings Moscow Volume 4, pp. 131-132, April 6-10, 2010.

[6] Nerses Safarayan, Hakob Sarukhanyan, "Learning Features By Means Of Genetic Programming For Object Recognition," Conference Computer Science and Information Technologies, 26 - 30 September, 2011, pp 382-385, Yerevan, Armenia.

[7] D. Howard, S. C. Roberts, and R. Brankin, "Target detection in SAR imagery by genetic programming," Advances in Engineering Software, Vol. 30, No. 5, pp. 303-311, Elsevier, May 1999.

[8] J. Shi and J. Malik. Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. and Machine Intell, 2000.

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

[10] Paul L. Rosin, Javier Hervas, "Image Thresholding For Landslide Detection By Genetic Programming," Proceedings of the First International Workshop on Multitemporal Remote Sensing Images, 13-14 Septembe, 2001 University of Trento, Italy