A Generic Structure of Object Classification using Genetic Programming

Arifa S Tamboli , Medha A Shah ,
Walchand College of Engineering, Sangli, MS (India),
shikalgar.arifa@gmail.com , shah.medha@gmail.com

***Abstract*—In This paper a method for classification of two types of objects using genetic programming (GP) has been presented. These two objects are coins of different sizes, and different textures. The basic algorithm of genetic programming was presented and explained. The features used for training and testing are mean, standard deviation, skewness and kurtosis. Precision and recall were used as performance measures and they were the main building blocks in building the fitness function. They replaced the false alarm and detection rate that was used in previous works. The result figures as well as values of precision, recall, fitness values, time elapsed, and number of generations used in training was presented. The very basic structure of a GP system was implemented and proved that it can work well as a standalone computational algorithm.**

***Index Terms*—Genetic Programming, GP, Object Classification, Object Recognition, Fitness function, Precision, Recall, Object Detection.**

I. INTRODUCTION

Object Classification (OC) is a topic which has wide spread use in various applications such as vehicle parking systems, object tracking, biometric applications such as face detection systems, optical character recognition, finger print recognition, classification of abnormalities in medical images, surveillance purposes and many computer vision applications. A variety of methods have been used for this purpose. Generative models such as principal component analysis [22], karhunen-loeve procedure [23] and eigenimages [24] were used for face recognition and characterization. A lot of other technologies like Hessian Matrix, Difference of Gaussian, Entropy Based Salient Region detectors etc. are used in many systems for object classification. Throughout all these systems, the basic steps remain the same which are first the training and then the testing stage. The most important step is method used for feature extraction. Features are the elementary units used in the representation of objects. These can be local features that measure metric properties of objects like measuring brightness or color, oriented lines, T-junctions, corners, etc. or global features that only represent qualitative characteristics of objects like 3D component parts realized as simple volumes that roughly capture the actual shape of an object. A good choice or combination of methods can give you a high precision. Various methods have been used in previous works such as wavelets [15] and [16]. In [25], back-propagation neural networks were used for character recognition ("Le Net"). In his SEEMORE model of object recognition, [26] employed a rich set of low-level features, including oriented edge filters, color filters, and blobs. Edge filter responses were combined into contours and corners. A neural network trained on the output of all these filters was able to recognize simple objects in photographs with good performance and, as expected, decreasing performance for degraded images. The use of moments as invariant binary shape representations was first proposed by Hu in 1961 [27]. Hu successfully used this technique to classify handwritten characters. Hence it can be seen that the basic steps include extraction of features, building the training model, and then testing it.

Genetic Programming (GP) which was introduced and explained by Koza [2] is an emerging technology which has been applied successfully to object recognition among other applications. Genetic Programming uses novel optimization techniques to "evolve" simple programs; mimicking the way humans construct programs by progressively re-writing them. It is actually derived from Genetic Algorithms (GA) [28]. GA's have been applied [19] - [21] along with neural networks for OC. GP has been applied to object classification by many researchers in combination with different features like mean and std. deviation [1], [5], [7]. Principal Components with GP was used in [8] for brain tumor classification. In another work, wavelets and GP was used with wavelets [15] for texture classification.

The remainder of this paper is organized as follows. Section II gives an overview of the basics of genetic programming. Section III describes our approach and experimental configuration towards achieving the above mentioned goals. Section IV presents the results of the experimentation. Section VI draws conclusions and gives future work directions.

II. GENETIC PROGRAMMING

Genetic algorithms contain a "population" of trial solutions

978-0-7695-4437-3/11 $26.00 © 2011 IEEE
DOI 10.1109/CSNT.2011.154

to a problem, typically each individual in the population is modeled by a string representing its DNA. This population is "evolved" by repeatedly selecting the "fitter" solutions and producing new solution from them. The new solutions replace existing solutions in the population. New individuals are created either asexually (i.e. copying the string) or sexually (i.e. creating a new string from parts of two parent strings).

In genetic programming the individuals in the population are computer programs. To ease the process of creating new programs from two parent programs, the programs are written as trees. New programs are produced by removing branches from one tree and inserting them into another. This simple process ensures that the new program is also a tree and so is also syntactically valid. The algorithm for a generic genetic programming problem is given as follows:

i. *Start*
ii. *Initialize population size, maximum iterations and other parameters*
iii. *Collect features to be evaluated*
iv. *Randomly initialize initial population*
v. *While (ideal program found or maximum no. of generations are completed)*
    a. *Evaluate fitness*
    b. *Select best programs to be inserted back into population (reproduction)*
    c. *Crossover operation using best programs*
        i. *Pairs of programs are randomly selected as parents.*
        ii. *They are "crossed-over" to generate two child programs.*
        iii. *If the child programs prove "fitter" than the parents, it is inserted into the population.*
    d. *Mutation operation using best programs*
        i. *Randomly select a program*
        ii. *Make a random change in the tree/ string.*
        iii. *If the mutated program proves "fitter" than the original, it is inserted into the population.*
vi. *End while*
vii. *Return best solution*
viii. *End*

A variety of representations exist for GP problems such as Tree based, Graph based, Cellular, Linear, and Grammar based etc. Tree based GP is the most commonly used [6]. There are few initialization techniques popular in tree based GP which are full method, grow method, ramped half-and-half method. The fitness measure assigns a fitness value to each possible fixed-length character string in the population. The fitness measure is often inherent in the problem. The fitness measure must be capable of evaluating every fixed-length character string it encounters. For each generation, the genetic algorithm first evaluates each individual in the population for fitness. Then, using this fitness information, the genetic algorithm performs the operations of reproduction, crossover, and mutation with the frequencies specified by the respective probability parameters pr, pc, and pm. This creates the new population.

III. METHODOLOGY

Many of the algorithms previously employed have used a sweeping window to collect the terminals. During the object classification process, a genetic program might consider many pixel positions in an image as object centers and we call each object centre localized in an image by a genetic program, a "classification position". The steps involved to find these positions are

1. Collect the terminal set from the data set. This indicates using the sweeping window to collect image statistics which are to be used for evaluation.
2. Evaluate these terminals using genetic program.
3. Calculate the fitness function based on that evaluation

A. TERMINAL SET

The terminals are typically either variable atoms (representing, perhaps, the inputs, sensors, detectors, or state variables of some system) or constants. Occasionally, the terminals are functions taking no explicit arguments, the real functionality of such functions lying in their side effects on the state of the system. The terminals we have used are mean, standard deviation, skewness and kurtosis. They are defined as follows:

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{N} \tag{1}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{N}} \tag{2}$$

$$Skewness = \sqrt{\frac{\sum_{i=1}^{n} (X_i - \bar{X})^3}{N\sigma^3}} \tag{3}$$

$$Kurtosis = \sqrt{\frac{\sum_{i=1}^{n} (X_i - \bar{X})^4}{N\sigma^4}} \tag{4}$$

where N = the total number of pixels in the window which are being considered,

X_i = the ith pixel and

$\bar{X}$ = the mean of pixels for i=1 to N

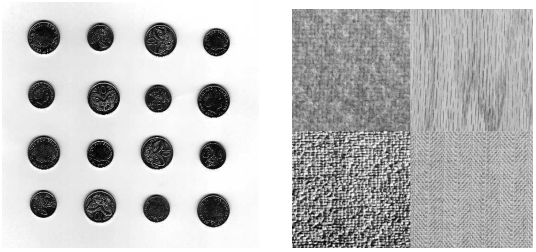
σ = the standard deviation of the pixels under the window.

B. FUNCTION SET

The function set is The set of operators used in a genetic program, e.g. '+', '-', '*', '/'. These act as the branch points in the parse tree, linking other functions or terminals. The function set we are using contains the four standard arithmetic operators and a conditional operation: FuncSet = {+, –, *, /, if}. The +, –, and * operators are usual addition, subtraction and multiplication, while / represents "protected" division. The *if* function returns its second argument if the first argument is positive or returns its third argument otherwise.

C. DATA SETS

We focus on classification of two types of data sets. One is distinguishing between 5c and 10c coins from coin images. The coins are located in different positions in these images with different orientations and appeared in different sides (head and tail). Three classes of 300 small objects (100 for each class) were cut out from the images and used to form the classification data set. The other is distinguishing between four different textures. The training images are shown below.



(a) (b)

Fig. 1. (a) Samples from data set consisting of 5c and 10c coins. (b) four textures used for training system

Fig. 2. Three classes used for training the data. (a) Heads, (b) Tails (c) background

The second set of images, Fig. 1(b) contains four different kinds of texture images, which are taken by a camera under the natural light. The images are taken from a web-based image database held by SIPI of USC [29]. The four texture classes are named woolen cloth, wood grain, raffia, and herringbone weave, respectively. This set has 400 sample cutouts from four large images, and each class has 100 samples.

D. CLASSIFICATION

For classification of the output as belonging to one class or the other, we use the output of the genetic program. The output of a genetic program in the standard GP system is a floating-point number. In the case of two classes "positive" or "negative", we can say that the output will lie on a number line and if the output is greater than 0, it is the positive class, else it belongs to the negative class. In the case of multiple classes such as the textures, various thresholds have to be set so as to account for the different classes such as in eqn. (5). T1 stands for threshold and should be set based on the object to be classified. The right thresholds can be found by trial and error. It has been depicted that two classes should have negative outputs and two positives. This is just a general case and not a requirement.

$$class = \begin{cases} \text{Class 1, } O < -T1 \\ \text{Class 2, } -T1 < O < 0 \\ \text{Class 3, } 0 < O < T1 \\ \text{Class 4, } O > T1 \end{cases} \quad (5)$$

E. FITNESS FUNCTION

We used precision and recall values to calculate fitness value. At the heart of the genetic programming problem lies the fitness function formulation. This function must give a measure of how well evolved the program is. This fitness function varies with each problem statement. The fitness function we are using was used for object localization using GP by Zhang and Malcolm Lett [10]. We use it for object recognition. They introduced the "Relative Localization Weighted F-measure" (RLWF), which attempts to acknowledge the worth of individual localizations made by the genetic program. Precision refers to the number of objects correctly recognized by a GP system as a percentage of the total number of object recognized by the system. Recall refers to the number of objects correctly recognized by a system as a percentage of total number of target objects in a data set. In this fitness function, instead of keeping the occurrence of a correct classification as "black or white", we will allow it have "shades of gray". This means we don't say 1=correct classification and 0=incorrect classification, but we use the scale [0,1] to signify how 'worthy' each classification is. We call this value as *classification fitness* CF. It is calculated based on the Euclidean distance between the current position and the correct position (closest object center). CF=1 indicates that the classification and CF=0 indicates it is not close to a correct classification position. CF can be calculated as follows:

$$CF(x,y) = \begin{cases} 1 - \dfrac{\sqrt{x^2+y^2}}{r}, if\ \sqrt{x^2+y^2} \le r \\ 0, otherwise \end{cases} \quad (6)$$

where $\sqrt{x^2+y^2}$ is the distance of the classification position (x,y) from target object centre, and r is called the "classification fitness radius", defined by the user. In this system, r is set to a half of the square size of the input window, which is also the radius of the largest object.

The precision and recall are calculated by taking the localization fitness for all the localizations of each object and dividing this by the total number of localizations or total number of target objects respectively.

$$\text{Precision}=\frac{Sum(\text{CF for all classification positions})}{\text{Total no. of classification positions}} \quad (7)$$

$$\text{Re}call=\frac{\text{Number of objects classified}}{\text{Total number of objects to be classified}} \quad (8)$$

$$Fitness=\frac{2X\ \text{Pr}ecisionX\ \text{Re}call}{\text{Pr}ecision+\text{Re}call} \quad (9)$$

Hence while calculating precision, for every classification position, a value between 1 and 0 will be used depending on how "precise" each classification is. This is divided by the total number of objects. Hence for a perfect case where each position represents the object at the exact center, precision value will be 1. Hence it is the measure of false alarms and trying to reduce those. However, this leaves a loophole. In the case where all object are not classified, one can still achieve precision 1. Hence recall value is used which checks for classification of all objects. Hence recall = 1 indicates that all objects are classified regardless of how good the precision is. Hence this measures detection ratio. Hence precision and recall replace false alarm rate and detection ratio that were used in [1]. Hence both these measures completely describe the fitness function which is given in eqn. (9).

F. PARAMETERS AND TERMINATION CRITERIA

The parameter values used in this approach are shown in table 1. In this approach, the learning/evolutionary process is terminated when one of the following conditions is met:

1) The detection problem has been solved on the training set, that is, all objects in each class of interest in the training set have been correctly detected with no false alarms. In this case, the fitness of the best individual program is zero.

2) The number of generations reaches the pre-defined number, max-generations.

TABLE 1
PARAMETER FOR GP TRAINING

	Parameters	Coin image	Texture image
Search Parameters	Population size	300	400
	Max. no. of generations	30	50
Genetic parameters	Reproduction rate	10%	10%
	Cross-rate	80%	70%
	Mutation-rate	10%	20%
Fitness parameters	Classification fitness radius	5	5

IV. RESULTS

In order to evaluate the system, along with the fitness function described above, we also use precision, recall and time taken. While implementing, after training and obtaining an evolved program, it was tested on the image and the output was displayed on a binary image with '0' for the background and '1' for the detected object. Since we try to minimize 'false positives' by maximizing precision, each 'object' should ideally be characterized by one classification position.

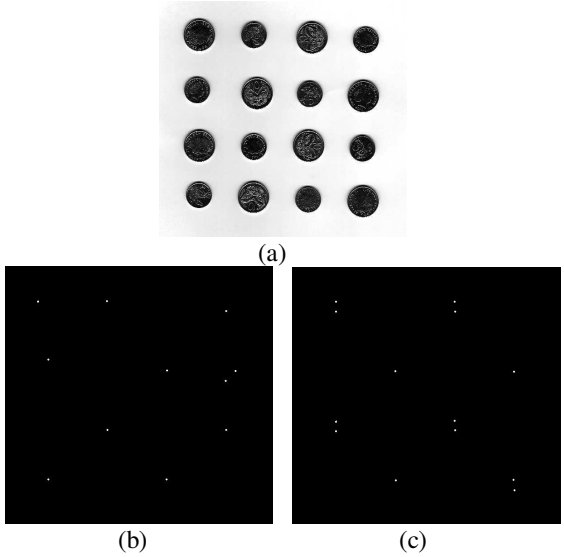


Fig. 3. (a) shows positions of 5c coins recognized from image (a) and (c) shows locations of 10c coins recognized from image (d)

Fig. 3(b) shows positions for 5c coins detected albeit with a few false positives while (c) shows positions of 10c coins recognized. Note that this is the case of two classes which is relatively easy. However, below is a case of four classes which have shown recognition of the 4 different textures.

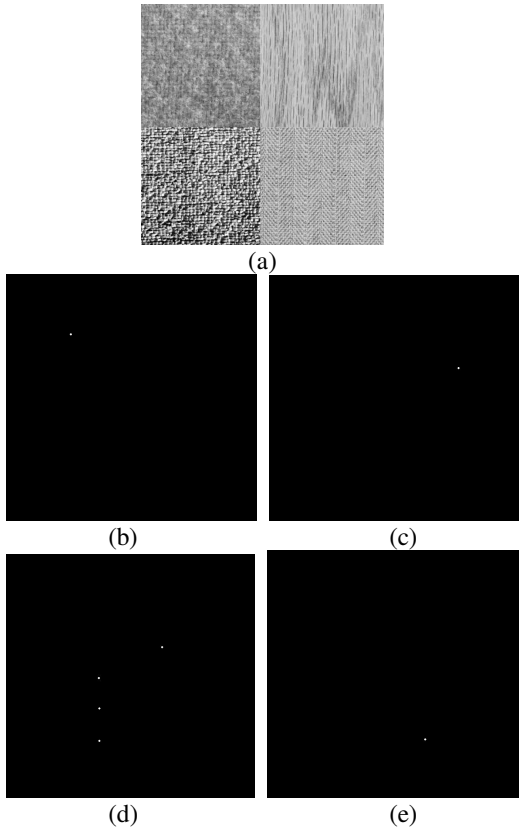


Fig. 4. (a) shows positions of 5c coins recognized from image (a) and (c) shows locations of 10c coins recognized from image (d)

TABLE 2
RESULTS OF THE GP SYSTEM

Data Set	Fitness	Test Accuracy		Training Efficiency	
		Precision	Recall	No. of Generations	Time taken (sec)
Coins	0.7413	0.589	1	21	11.2414
Texture	0.877	0.782	1	35	24.8361

V. CONCLUSIONS

The objective was to prove the potential of GP as a tool for object classification. By taking two datasets of coins and textures, we have collected terminals of mean, standard deviation, skewness and kurtosis. Inspired by the concept of survival of the fittest and natural selection described by Charles Darwin in '*On the Origin of Species by Means of Natural Selection*', these terminals were used to evaluate many generations of evolved genetic programs to give us the fittest program. Results presented give the efficiency and accuracy of the system. Here GP has been used alone without another machine learning technology and yet gives good results. Precision & Recall have proved as excellent performance measures and have replaced false alarm and detection rates. Genetic programming has proved that it can be used standalone as an efficient object recognition engine.

VI. REFERENCES

[1] Chandrashekhar Padole, Joanne Athaide, "Genetic Programming for Object Detection: Optimizing Terminal Set Acquisition and Fitness Function", Proc. Intl. Conf. Signal, Systems and Automation, 2011

[2] J. R. Koza, "Genetic programming: on the programming of computers by means of natural selection". London, England: Cambridge, Mass.: MIT Press, 1992.

[3] A. Song, V. Ciesielski, and H. Williams, "Texture classifiers generated by genetic programming," in Proc. CEC, D. B. Fogel, M. A. El Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, andM. Shackleton, Eds., 2002, pp. 243–248.

[4] T. Loveard and V. Ciesielski, "Representing classification problems in genetic programming," in Proc. Congr. Evol. Comput., May 27–30, 2001, vol. 2, pp. 1070–1077.

[5] S. A. Stanhope and J. M. Daida, "Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery," in Proc. 7th Annu. Conf. Evol. Program., V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds., San Diego, CA, Mar. 25–27, 1998, Lecture Notes in Computer Science, vol. 1447, pp. 735–744.

[6] Hajira Jabeen and Abdul Rauf Baig, "Review of Classification Using Genetic Programming" International Journal of Engineering Science and Technology Vol.2 (2), 2010, 94-103

[7] Mengjie Zhang, Member, IEEE, Xiaoying Gao, and Weijun Lou, 'A New Crossover Operator in Genetic Programming for Object Classification' IEEE TRANSACTIONS on Systems, man, and cybernetics—Part B: Cybernetics, Vol. 37, NO. 5, October 2007

[8] H. F. Gray, R. J. Maxwell, I. Martinez-Perez, C. Arus, and S. Cerdan, "Genetic programming for classification of brain tumours from nuclear magnetic resonance biopsy spectra," in Proc. 1st Annu. Conf. Genetic Program., J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds., Jul. 28–31, 1996, p. 424.

[9] R. Poli, "Genetic programming for image analysis," in Proc. 1st Annu. Conf. Genetic Program., J. R. Kosa, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds., Jul. 28–31, 1996, pp. 363–368.

[10] Mengjie Zhang, Malcolm Lett. "Genetic Programming for Object Detection: Improving Fitness Functions and Optimising Training Data". IEEE Intelligent Informatics Bulletin (IEEE Computational Intelligence Bulletin). Vol. 7, No. 1. 2006. pp. 12-21.

[11] J. F. Winkeler and B. S. Manjunath, "Genetic programming for object detection," in Proc. 2ndAnnu. Conf. Genetic Program., J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, Eds., Jul. 13–16, 1997, pp. 330–335.

[12] A. Teller andM. Veloso, "A controlled experiment: Evolution for learning difficult image classification," in Proc. 7th Portuguese Conf. Artif. Intell., C. Pinto-Ferreira and N. J. Mamede, Eds., Oct. 3–6, 1995, Lecture Notes in Artificial Intelligence, vol. 990, pp. 165–176.

[13] M. Zhang and V. Ciesielski, "Genetic programming for multiple class object detection," in Proc. 12th Australian Joint Conf. AI, N. Foo, Ed., Dec. 1999, Lecture Notes in Artificial Intelligence, vol. 1747, pp. 180–192.

[14] Y. Le Cun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in Intelligent Signal Processing. Piscataway, NJ: IEEE Press, 2001, pp. 306–351.

[15] Zheng Chen; Siwei Lu; "A Genetic Programming Approach for Classification of Textures Based on Wavelet Analysis" IEEE International

Symposium on Intelligent Signal Processing, 2007. WISP 2007. Publication Year: 2007 , Page(s): 1 - 6

[16] M. R. Azimi-Sadjadi, D. Yao, Q. Huang, and G. J. Dobeck, "Underwater target classification using wavelet packets and neural networks," IEEE Trans. Neural Netw., vol. 11, no. 3, pp. 784–794, May 2000.

[17] Ji H. Moon, Sung G. Lee, Sung Y. Cho, and Yoo-Sung Kim, "A Hybrid On-Line Signature Verification System Supporting Multi-Confidential Levels Defined by Data Mining Techniques", 6th International Conference on Information Technology and Applications (ICITA 2009)

[18] W. A. Tackett, "Genetic programming for feature discovery and image discrimination," in Proc. 5th ICGA, S. Forrest, Ed., Jul. 17–21, 1993, pp. 303–309.

[19] P. G. Korning, "Training neural networks by means of genetic algorithms working on very long chromosomes," Int. J. Neural Syst., vol. 6, no. 3, pp. 299–316, Sep. 1995.

[20] V. Ciesielski and J. Riley, "An evolutionary approach to training feed forward and recurrent neural networks," in Proc. 2ndInt. Conf. Knowledge-Based Intell. Electron. Syst., L. C. Jain and R. K. Jain, Eds., Adelaide, Australia, Apr. 1998, pp. 596–602.

[21] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," IEEE Trans. Neural Netw., vol. 8, no. 3, pp. 694–713, May 1997. [Online]. Available: citeseer.ist.psu.edu/yao96new.html

[22] Hui Kong, LeiWang, Eam Khwang Teoh, Xuchun Li, Jian-GangWang, and Ronda Venkateswarlu. Generalized 2d principal component analysis for face image representation and recognition. In Proc. IEEE Joint Intern. Conf. on Neural Networks, volume I, pages 108{113, 2005.

[23] Michael Kirby and Lawrence Sirovich. Application of the karhunen-loeve procedure for the characterization of human faces. IEEE Trans. on Pattern Analysis and Machine Intelligence, 12(1):103{108, 1990.

[24] Ale·s Leonardis and Horst Bischof. Robust recognition using eigenimages. Computer Vision and Image Understanding, 78(1):99{118, 2000.

[25] LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE pp. 2278–2324.

[26] Mel BW (1997) SEEMORE: combining color, shape, and texture histogramming in a neurally inspired approach to visual object recognition. Neural Computation 9:777–804.

[27] Hu, M. K., \Visual Pattern Recognition by Moment Invariants", IRE trans. Information Theory, Vol. 8, pp. 179-187.

[28] John H. Holland. Adaptation in Natural and Arti_cial Systems: An Introductory Analysis with Applications to Biology, Control and Arti_cial Intelligence. MIT Press, 1992. First Published by University of Michigan Press 1975.

[29] Signal and Image Processing Institute of University of Southern California. accessed on Jul. 22, 2004. [Online]. Available: http://sipi.usc.edu/services/database/database.cgi?volume=textures