ISSN (Online): 2230-7893
www.IJCEM.org

Time bound Adaptive Genetic Algorithm based face recognition

S. N. Palod[1], S. K. Shriwastava[2], P. K. Purohit[3]

[1]Smt. Radhikatai Pandav College of Engineering, Nagpur, India
[2]Shri Balaji Institute of Technology & Management, Baitul (M.P), India
[3]National Institute of Technical Teachers Training & Research, Shamla Hills, Bhopal, India

Abstract

When huge face database has to be searched for face detection time becomes the deciding factor for certain applications such as airport security checks where face detection within a short span of time is desirable. Genetic algorithm helps in close match in a very large face database based on heuristic search. PCA based features is a compulsion from the point of view of low dimension and space limitation. This paper proposes time bound adaptive genetic algorithm so as to reduce time for face detection application in shortest span of time. Integrated expert system will still reduce time for recognition.

Keywords : Genetic Algorithm, PCA, ICA.

I. INTRODUCTION

Face recognition is a biometric authentication method that has become more significant and relevant in recent years. It is becoming a more mature technology that has been employed in many large scale systems such as Visa Information System, surveillance access control and multimedia search engine. Generally, there are three categories of approaches for recognition, namely global facial feature, local facial feature and hybrid feature. Although the global facial-based feature approach is the most researched area, this approach is still plagued with many difficulties and drawbacks due to factors such as face orientation, illumination, and the presence of foreign objects.

The genetic algorithm was introduced by Regensburg in 1990. (GAs) is a class of optimization procedures inspired by the mechanisms of natural selection [1, 2]. (GAs) operates iteratively on a population of structures, each of which represents a candidate solution to the problem, encoded as a string of symbols (chromosome). A randomly generated set of such strings forms the initial population from which the (GAs) starts its search. Three basic genetic operators guide this search: selection, crossover and mutation.

The basic concept behind Genetic & Evolutionary Computation (GEC) is to find an optimal (or near optimal) solution for a specific problem [5, 6, 7]. First, a number of individuals or candidate solutions are generated to form an initial population. Each individual is then evaluated and assigned a fitness obtained from the evaluation function specific to the problem at hand. Parents are then selected and new individuals are produced from the selected parents by the processes of reproduction. Survivors are selected from the previous generation and combined with the offspring to form the next generation. This process continues for user specified number of cycles..

PCA has been called one of the most valuable results from applied linear algebra. is used abundantly in all forms of analysis (from neuroscience to computer graphics) because it is a simple, nonparametric method of extracting relevant information from confusing datasets. With minimal additional effort, PCA provides a roadmap for how to reduce a complex dataset to a lower dimension to reveal the sometimes hidden, simplified structure that often underlie it [3].

II. Genetic Algorithm methodology

GA is a powerful search and optimization algorithm, which are based on the theory of natural evolution. In GA, each solution for the problem is called a chromosome and consists of a linear list of codes. The GA sets up a group of imaginary lives having a string of codes for a chromosome on the computer. The GA evolves the group of imaginary lives (referred to as population), and gets and almost optimum solution for the problem. The GA uses three basic operators to evolve the population: selection, crossover, and mutation.

Genetic algorithm was developed by John Holland-University of Michigan (1970s) to provide efficient techniques for optimization and machine learning applications through application of the principles of evolutionary biology to computer science. It uses a directed search algorithms based on the mechanics of biological evolution such as inheritance, mutation, natural selection, and recombination (or crossover). It is a heuristic method that uses the idea of survival of the fittest. In the genetic algorithm, the problem to be solved is represented by a list of parameters which can be used to

drive an evaluation procedure, called chromosomes or genomes. Chromosomes are typically represented as simple strings of data and instructions. In the first step of the algorithm, such cromosomes are generated randomly or heuristically to form an initial pool of possible solutions called *first generation pool*. In each generation, each organism (or individual) is evaluated, and a value of *goodness* or *fitness* is returned by a fitness function. In the next step, a second generation pool of organisms is generated, by using any or all of the genetic operators: selection, crossover (or recombination), and mutation. A pair of organisms are selected for to survive based on elements of the initial generation which have better fitness. In other words, the organisms that have relatively higher fitness than other organisims in the generation are selected to survive. Some of the well-defined organism selection methods are roulette wheel selection and tournament selection. After selection, the *crossover* (or *recombination*) operation is performed on the selected chromosomes, with some probability of crossover *(Pc)*- typically between 0.6 and 1.0. Crossover results in two new child chromosomes, which are added to the second generation pool. The crossover operation is done by simply swapping a portion of the underlying data structure of the chromosomes of the parents. This process is repeated with different parent organisms until there are an appropriate number of candidate solutions in the second generation pool. In the mutation step, the new child organism's chromosome is randomly mutated by randomly altering bits in the chromosome data structure. The aim of these is to produce a second generation pool of chromosomes that is both different from the initial generation and hence have better fitness, since only the best organisms from the first generation are selected for surviving. The same process is applied for the second, third, generations until an organism is produced which gives a solution that is "good enough".
The genetic algorithm and different genetic operations are shown by a flowchart shown in next pages.

The algorithm starts with an initial set of random solutions called the population. Each individual in the population, known as chromosome, represents a particular solution of the problem. Each chromosome is assigned a fitness value depending on how good its solution to the problem is.

After fitness allotment, the natural selection is executed and the 'survival of the fittest chromosome' can prepare to breed for the next generation. A new population is then generated by means of genetic operations: cross-over and mutation. This evolution process is iterated until a near-optimal solution is obtained or a given number of generations is reached.

Fitness function

In order to identify the best individual during the evolutionary process, a function needs to assign a degree of fitness to each chromosome in every generation. So in order to determine whether the assumed region of the input image is a face or not, the fitness value of the possible face region is computed by means of similarity.

Selection:

Selection operator is a process in which chromosomes are selected into a mating pool according to their fitness function. Good chromosomes that contribute their gene-inherited knowledge to breed for the next generation, are chosen.

Cross-over:

This operator works on a pair of chromosomes and produces two offsprings by combining the partial features of two chromosomes. Here we will have to learn single point cross-over, two point cross-over and uniform cross-over operators.

Mutation:

This operator alters genes with a very low probability.

There are 6 parameters to control the operation of the Genetic Algorithm:

Population Size - The population size is the initial number of random tours that are created when the algorithm starts. A large population takes longer to find a result. A smaller population increases the chance that every tour in the population will eventually look the same. This increases the chance that the best solution will not be found.

Neighborhood / Group Size - Each generation, this number of tours are randomly chosen from the population. The best 2 tours are the parents. The worst 2 tours get replaced by the children. For group size, a high number will increase the likelihood that the really good tours will be selected as parents, but it will also cause many tours to never be used as parents. A large group size will cause the algorithm to run faster, but it might not find the best solution.

Mutation % - The percentage that each child after crossover will undergo mutation. When a tour is mutated, one of the cities is randomly moved from one point in the tour to another.

Nearby Cities - As part of a greedy initial population, the GA will prefer to link cities that are close to each other to make the initial tours. When creating the initial population this is the number of cities that are considered to be close.

Nearby City Odds % - This is the percent chance that any one link in a random tour in the initial population will prefer to use a nearby city instead of a completely random city. If the GA chooses to use a nearby city, then there is an equally random chance that it will be any one of the cities from the previous parameter.

Maximum Generations - How many crossovers are run before the algorithm is terminated.

The other options that can be configured are (note: these are only available in the downloadable version):

Random Seed - This is the seed for the random number generator. By having a fixed instead of a random seed, you

can duplicate previous results as long as all other parameters are the same. This is very helpful when looking for errors in the algorithm.

III. Adaptive Approach to GA

In contrast to simple genetic algorithms (SGA), the researchers have recently been proposing new adaptive approaches in the GA for both the penalty function and the mutation and crossover operators to increase the probability of capturing the global optimum, to enhance the performance of the GA and to relieve the user from the burden of having to determine sensitive parameter(s) [12,18–22]. In this study, a new initial population strategy based adaptive approach considering time as parameter for decisive factor for size of initial population. The time available for face detection from a huge database will be a user specific option. While approach will take care of deciding size of population in GA. In this way time will play a important role since user does not have to try all possible solutions to find global minima or maxima which is a foundation of Genetic Algorithm. Population sizing has been one of the important topics to consider in evolutionary computation [1], [13]. Researchers usually argue that a "small" population size could guide the algorithm to poor solutions [12], [13],[11] and that a "large" population size could make the algorithm expend more computation time in finding a solution, [09], [11]. So, here we are facing a trade-off that needs to be approximated feeding the algorithm with "enough" chromosomes [19], in order to obtain "good" solutions. "Enough," for us, is directly related to instances in the search space and diversity.

It has been recognized that if the initial population to the GA is good, then the algorithm has a better possibility of finding a good solution [4], [21] and that, if the initial supply of building blocks is not large enough or good enough, then it would be difficult for the algorithm to find a good solution. Sometimes, if the problem is quite difficult and some information regarding the possible solution is available, then it is good to seed the GA with that information [5], [16], i.e., the initial population is seeded with some of those possible solutions or partial solutions of the problem. A measure of diversity plays a role here in the sense that, when we have no information regarding a possible solution, then we could expect, that the more diverse the initial population is, the greater the possibility to find a solution is, and of course, the number of individuals in the population to get a good degree of diversity becomes important.

IV. PCA Based Feature Extraction

Principal component analysis, from a statistical perspective, is a method for (i) transforming correlated variables into uncorrelated variables, (ii) finding linear combinations of the original variables with relatively large or small variability, and (iii) reducing data. The PCA was developed by Pearson (1901), primarily as a tool for reducing data (Flury 1988). Here, we use the PCA as a method that reduces data dimensionality by performing a covariance analysis between factors. As such, it is suitable for data sets in multiple dimensions.

For feature extraction suppose N x N pixel 2D images of faces. Each image is considered to be a single point in face space (a single sample from a random vector) and each dimension (each pixel) can take real values between 0 and 255. Let be the n sample images. Find the mean image, m, and subtract it from each image: Zi Xi - m. Let A be the matrix whose columns are the mean-subtracted sample images.

V. Result

Our training set contains 471 faces and 500 non-faces sub images which were extracted manually from a face image dataset. Each sub image in the training and test sets was scaled to same size and preprocessed to account for different lighting conditions and contrast.

No. of Test Images	Time Required for Face Detection (classical GA)	Accuracy of Detection (classical GA)	Time Required for Face Detection (Time Bound GA)	Accuracy of Detection (Time Bound GA)
100	10	80 %	10	80 %
1000	10	70 %	12	78 %
100000	10	35 %	25	52 %
1000000	10	5 %	45	22 %

CONCLUSIONS

Time bound limitation is realistic for real time practical applications of face detection. This paper adds a significant parameter "time" to decide the population of initial population to be selected randomly. Hence genetic algorithm picks up adaptive nature and will always yield close to final detection in a bound time span with improved accuracy of detection. Results also revealed the same. Expert system added together with genetic can increase the confidence of face detection.

REFERENCES

[1] D. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, 1989.
[2] J. Holland, Adaptation in Natural and Artificial Systems, MIT Press, 1992.
[3] J.shlens, "A Tutorial on Principle Component Analysis," reported by systems Neurobiology Laboratory, Salk Institute for biological studies La Jolla, CA 92037 and Institute for Nonlinear Science, University of California, San Diego L Jolla, CA 92093-0402, December 10,2005.
[5] L. Davis, *Handbook of Genetic Algorithms . New York: Van _ostrand Reinhold, 1991.*
[6] D. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press, 2000.*
[7] J.Adams, D. Woodard, G. Dozier, P. Miller, G. Glenn, and K. Bryant, "GEFE: Genetic & evolutionary feature extraction for periocular-based biometric recognition," in *Submission to: The ACM Southeastern Conference2010*. New York, NY, USA: ACM, 20.
[8] L. Wiskott, J. Fellous, N. Kruger, and C. von der Malsburg, "Face Recognition by Independent Component Analysis",, IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 13, NO. 6, NOVEMBER 2002.
[9] D. Zhang and Z. Wangmeng. "Computational intelligence-based biometric *technologies*." Computational Intelligence Magazine, IEEE, 2(2):26{36, May 2007.
[10] C. Liu, H. Wechsler, "*Evolutionary pursuit and its application to face recognition"*, IEEE Trans. Patt.Anal. Mach. Intell. 22 (6) (2000) 570–582.
[11] Zhao, Q., Zhang, D., Zhang, L., and Lu, H. "*Evolutionary discriminant feature extraction with application to face recognition"*. EURASIP Journal on Advances in Signal Processing.,2009
[12] Togn V, Daloglu AT. Optimization of 3d trusses with adaptive approach in genetic algorithms. Eng Struct 2006;28:1019‐27.
[13] Togan V, Daloglu AT. Adaptive approaches in genetic algorithms to catch the global optimum. In: Proc 7th international congress advances in civil engineering, Istanbul; 2006. p. 244.
[14] Krishnamoorthy CS, Venkatesh PP, Sudarshan R. Object-oriented framework for genetic algorithms with application to space truss optimization. J Comput Civil Eng, ASCE 2002;16:66‐75.
[15] Sudarshan R. Genetic algorithms and application to the optimization of space trusses, A Project Report, Madras (India), Indian Institute of Technology; 2000.
[16] American Institute of Steel Construction (AISC). Manual of steel construction-allowable stress design. 9th ed., Chicago; 1989.
[17] Turkish Standard Institute (TSE). Building code for steel structures TS 648, Ankara, Turkey; 1982.
[18] Kaveh A, Kalatjari V. Size/geometry optimization of trusses by the force method and genetic algorithm. Z Angew Math Mech 2004;84(5):347‐57.
[19] Nanakorn P, Meesomklin K. An adaptive penalty function in genetic algorithms for structural design optimization. Comput Struct 2001;79:2527‐39.
[20] Barbosa HJC, Lemonge ACC. A new adaptive penalty scheme for genetic algorithms. Inform Sci 2003;156:215‐51.
[22] Coello CAC. Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 2000;41:113‐27.
[23] Coit DW, Smith AE, Tate DM. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. INFORMS J Comput 1996;8(2):173‐82.