PalmPrints: a Cooperative Co-Evolutionary Algorithm for Clustering Hand Images

N. Kharma[1], *Member, IEEE*, C. Y. Suen, *Fellow, IEEE,* P. F. Guo

Abstract—**The main objective of Project PalmPrints is to develop and demonstrate a special co-evolutionary genetic algorithm (GA) that optimizes (a clustering fitness function) with respect to three quantities, (a) the dimensions of the clustering space; (b) the number of clusters; and (c) and the locations of the various clusters. This genetic algorithm is applied to the specific practical problem of hand image clustering, with success. In addition to the above, this research effort makes the following contributions: (a) a CD database of (raw and processed) right-hand images; (b) a number of novel features designed specifically for hand image classification; (c) an extended fitness function, which is particularly suited to a dynamic (i.e. dimensionality varying) clustering space. Despite the complexity of the multi-optimizational task, the results of this study are clear. The GA succeeded in achieving a maximum fitness value of 99.1%; while reducing the number of dimensions (features) of the space by more than half (from 84 to 41).**

Index Terms—hand, image processing, pre-processing, feature selection, pattern clustering, genetic algorithms, cooperative co-evolution.

I. INTRODUCTION

Clustering is the grouping of similar objects (e.g. hand images) together in one set. It is an important unsupervised classification technique. In it, a set of patterns is partitioned such that, according to some criterion, patterns in any one partition are similar to each other, and dissimilar to the patterns belonging to the other partitions. Several clustering techniques are available in the literature: branch and bound procedures [5], maximum likelihood estimation techniques [6] and graph theoretic approaches [7]. The simplest and most well known clustering algorithm is the k-means algorithm [8]. However, this algorithm requires that the user specifies, before hand, the desired number of clusters.

A GA-based clustering technique was proposed in [18]. In it, the GA was used to optimize cluster centres, though the number of clusters was fixed before hand. It did nevertheless; demonstrate that a GA-based clustering algorithm can provide better performance than the k-means algorithm. In [10], another genetic algorithm was used to determine the best number of clusters, while simultaneously clustering objects. Also, in [3], a hybrid genetic algorithm for feature selection resulted in (a) better convergence properties; (b) significant improvement in performance; and (c) reduction in the number of features used. In [4], a GA, in combination with a k-nearest neighbour classifier, was successfully employed for feature reduction. In [19] the author presented a GA for clustering objects into a given number of similarity classes. The algorithm was evaluated using test sets, and found to be very efficient as well as robust in the presence of noise or imprecisions. All of these studies demonstrate that genetic algorithms can act effectively to (a) reduce features; (b) decide the number of clusters; and (c) determine the locations of the clusters. None of them, however, tackle all three objectives simultaneously with a single evolutionary or co-evolutionary algorithm.

Cooperative co-evolution refers to the simultaneous evolution of two (or more) species with coupled fitnesses. Such evolution allows the discovery of complex multi-module solutions with low coupling between the modules. Cooperative co-evolutionary algorithms involve a number of independently evolving populations of (individuals or) part-solutions to some problem. The fitness of an individual, in any of these populations, depends on its ability to collaborate with individuals from the other populations to offer a

[1] Corresponding author N. Kharma is with the Electrical & Computer Engineering Dept., Concordia University, 1455 de Maisonneuve Blvd. West, Montreal, QC, H3G 1M8, Canada; telephone: 514-848-2424 ext. 3117; fax: 514-848-2802; e-mail: kharma@ece.concordia.ca.

complete solution to the whole problem [12].

The problem is to divide a set of hand images into a number of clusters without *a priori* knowledge of the feature space. And, the solution that this paper proposes is a cooperative co-evolutionary clustering algorithm, which integrates dynamic clustering, with (hand-based) feature selection.

This paper is organized as follows. In PART II, the database of hand images is introduced. In PART III, hand feature extraction is described. In PART IV, the co-evolutionary clustering algorithm, including the modified fitness function is presented. PART V provides implementation results and final conclusions.

II. DATA ACQUISITION AND PREPROCESSING

All hand images were acquired using a Scanjet 3P scanner from Hewlett Packard™. Images were converted from Bitmap to JPEG format using 20/20 from byLight Technologies ™ [13].

Insert Figure 1 Here.

A. Image Enrolment

During the enrolment phase, a user placed his/her right hand on the glass for scanning, hand facing downward. Five images of the same hand were taken in succession. The user removed his/her hand completely from the glass between scans, resulting in 5 slightly different images of his/her hand. Scans were taken at a resolution of 439 x 319 pixels, in Black and White Photo mode. Each hand takes 3 seconds to be scanned once.

B. Pre-processing

Pre-processing consists of three steps, (a) image binarization; (b) application of a median filter; and (c) image contourization. This process is shown in Fig.2.

Insert Figure 2 Here.

1) Image Binarization: The purpose of image binarization is to divide the pixels of an image into either a 'black' group or a 'white' group. If the value of a pixel is less than or equal to a threshold, the pixel is assigned to the 'white' group, or else it is assigned to the 'black' group.

2) Median Filter: The purpose of this filter is to remove white noise and hence enhance the clarity of the hand contour.

3) Image Contourization: Hand contourization is a transfer function that takes a greyscale image of a hand and produces a smooth continuous border of the hand. These contours are binary in nature.

For a pixel to qualify as being part of the hand, it must be surrounded by white pixels. The top, bottom, left and right pixels are checked, one by one, to decide the contour. The resulting image contour is shown in Fig.3.

C. Pre-processing Results And Discussion

Two hundred different people contributed to the database, each with 5 images of his/her right hand. Of the 1000 images, about 83% showed 'good' contour output. The problems associated with the remaining images were predominantly the results of poor system usage. Therefore, it is our belief that the establishment of a few user guidelines (not necessarily explicit) would significantly improve the quality of captured hand images, regardless of the intended application.

Insert Figure 3 Here.

The outputs include both (a) images of the resulting contours, as well as (b) text files representations of the contours. A text file holds the sequence of (x,y) coordinates of the points making up the contour. image and a text file, which represents the contour dimension values in the x-y plane. All raw input images and processed output results are stored on the CD database, which we use for this study.

III. FEATURE EXTRACTION

A human hand can be treated as a rigid body of definite shape and geometrical parameters; hand geometry refers to the geometric structure of the hand. Shape analysis requires the extraction of object features, often normalized, and invariant to various geometric transformations, such as translation, rotation and scaling.

Insert Figure 4 Here.

A. Finger Reorientation Algorithm

Features play a central role in image recognition. There are three goals implicit in the design of pattern classifiers [14]:

1) *Optimization of classification accuracy;*
2) *Minimization of feature extraction cost; and*
3) *Reliable performance.*

Since each finger of the hand is place-free when scanned, an adaptive method for finger reorientation is applied. The purpose is to vertically align each finger prior to feature extraction.

B. Reorientation Algorithm

The tip of a finger T point, together with the minima of the two phalanges at either side of that finger A and B, make up a triangle ΔTAB. The point C of that triangle is defined as the finger centre point.

Insert Figure 5 Here.

The line $\overline{AB}$ connecting the minima of the two phalanges A and B is termed the finger base line. This line has a midpoint: call it finger base-line midpoint, or Om. The line $\overline{CO_m}$ connecting Om with the finger centre point C makes an angle θ with the vertical Y-axis, shown in Fig.6. This is the angle by which the finger is rotated in order to make the finger stand upright. Each finger (say finger *i*) is rotated by its own rotation angle θ*i*. This whole process is depicted in Fig.7; it is also described by the following formulae:

$$X'' = X' \cos\theta - Y' \sin\theta = (X - Xm)\cos\theta - (Y - Ym)\sin\theta \quad (1)$$

$$Y'' = X' \sin\theta + Y' \cos\theta = (X - Xm)\sin\theta + (Y - Ym)\cos\theta \quad (2)$$

X : the principal x-axis
X': the x-axis after the first transformation
X'': the x-axis after the second (and final) transformation
Y : the principal y-axis
Y': the y-axis after the first transformation
Y'': the y-axis after the second (and final) transformation
Om (Xm,Ym): the finger base-line mid point
O(Xo,Yo): the origin of the original system of coordinates
θ : the rotation angle (If the plane is rotated clockwise, $\theta \geq 0$. Otherwise, $\theta < 0$)

Insert Figure 6 Here.

C. Reorientation Results

The performance of the finger reorientation algorithm (shown in Fig. 8 and Table II) is analyzed via the computation of the Coefficient of Variance (CV). Smaller values of CV represent less variability to the mean [15]. CV is calculated over the various images of the same hand before, as well as after, reorientation.

The finger reorientation experimental results (shown in Table III) clearly show that reorientation consistently led to more stable feature values, *regardless* of the features measured. This is a significant find. Hence, it is highly recommended that any feature extraction be carried out, only after the various fingers have been made to stand upright.

Insert Figure 7 Here.

D. Geometric Features Extraction

The geometrical features measured can be divided into six categories:

- *Finger Width:* the distance between the minima of the two phalanges at either side of a finger.
- *Finger Height:* the length of the line starting at the fingertip and intersecting (at right angles) with finger base-line.
- *Finger Circumference:* The length of the finger contour.
- *Finger Angle*: The two acute angles made between the finger base-line and the two lines connecting the phalange minima with the finger tip.
- *Finger Base Length*: The length of the finger base-lines.
- *Palm Aspect Ratio*: the ratio of the 'palm width' to the 'palm height'. Palm width is (double) the distance between the midpoint of the base-line of the middle finger, and the midpoint of the line connecting the outer points of the base lines of the thumb and pinkie (call it Mp). Palm length is (double) the shortest distance between Mp and the right edge of the hand image, shown in Fig. 8.

Insert Figure 8 Here.

E. Statistical Features Extraction

Each finger has its own characteristics in hand shape analysis. Before doing shape analysis, we map each 2-D finger contour onto 1-D contour space (inspired by, but still significantly different from [16]), taking the finger midpoint center as its reference point. The shape analysis for four fingers, excluding the thumb, was measured after the finger reorientation procedure using: Central Moments; Fourier Descriptors; and Zernike Moments.

1) *Finger 1-D Contour Transformation:* Once the fingers are reoriented, such that each finger is separate and vertically aligned, a 2D-to-1D mapping is carried out. The 2D contour of each finger is mapped onto a 1D graph (see Fig. 9). The horizontal axis represents the index of a point on the 2D contour of a finger. The y-axis represents the distance between the contour point and the centre of the finger. (The centre of a finger is defined as the centre of the triangle with the tip of the finger as one vertex and the two ends of the finger's base-line as the other two vertices). Each and every finger is represented by the same number of points: 140. Those points are split evenly into two sets: Seventy to the left of the tip of the finger and another seventy to the right of it. These points are represented by F[n] with point index $n \in [0, 140]$. F[n] is defined as the Euclidean distance between point (with index) n and the reference point.

Insert Figure 9 Here.

Assuming that the obtuse angle corresponding to the 1-D contour curvature is divided into equal angle

segments for the total 140 points, the length of the vector between the reference point and index can be described in polar form as $f[\rho_j, \varphi_j]$ for the j_{th} pixel.

2) *Central Moments:* For a digital image, the p_{th} order regular moment with respect to a one-dimensional function F[n] is defined as follows:

$$R_p = \sum_{n=0}^{N} n^p \cdot F[n] \quad (3)$$

The normalized one-dimensional p_{th} order central moments are defined as follows:

$$M_p = \sum_{n=0}^{N} (n-\bar{n})^p \cdot F[n] \quad (4)$$

$$\bar{n} = R_1 / R_0 \quad (5)$$

F[n]: with n∈[0,N]; the sequence used for representing each finger contour in 1-D whose values equal the distance between point n and the finger reference point
N: the total number of pixels

3) *Fourier Descriptors:* We may define a normalized cumulative function Φ* as an expanding Fourier series to obtain descriptive coefficients (termed Fourier Descriptors or FD's). Given a periodic 1-D digital function F[n] in [0, N] points (periodic), the expanding Fourier series is:

$$\Phi^*(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos\frac{2\pi k}{N} \cdot t + b_k sin\frac{2\pi k}{N} \cdot t\right) \quad (6)$$

$$a_k = \frac{2}{N}\sum_{n=1}^{N} F[n] \cdot \cos\frac{2\pi k}{N} \cdot n \quad (7)$$

$$b_k = \frac{2}{N}\sum_{n=1}^{N} F[n] \cdot sin\frac{2\pi k}{N} \cdot n \quad (8)$$

The k_{th} harmonic amplitudes of the Fourier Descriptors are:

$$A_k = \sqrt{a_k^2 + b_k^2} \quad k = 1,2,3, .. \quad (9)$$

F[n]: with n∈[0,N] be the sequence used for representing each finger contour in 1-D whose values equal the distance between point n and the finger reference point
N: the total number of pixels

4) *Zernike Moment:* For a digital image with a polar form function $f(\rho(\varphi))$, the normalized $(n+m)_{th}$ order Zernike moments are approximated by:

$$Z_{nm} \approx \frac{n+1}{N}\sum_j f(\rho_j, \varphi_j) \cdot V_{nm}^*(\rho_j, \varphi_j) \quad (10)$$

$$V_{nm}(\rho, \varphi) = R_{nm}(\rho) \cdot e^{jm\varphi} \quad (11)$$

with $x_j^2 + y_j^2 \le 1$

$$R_{nm}(\rho) = \sum_{s=0}^{(n-|m|/2)} \frac{(-1)^s (n-s)! \rho^{n-2s}}{s!((n+|m|)/2-s)!((n-|m|)/2-s)!} \quad (12)$$

$f(\rho_j, \varphi_j)$: the pixel value of the j_{th} pixel
n: a positive integer
m: a positive or negative integer subject to the constraints that n-|m| is even, and |l|≤ n.
N: the total number of pixels
*: denotes the complex conjugate

IV. Simultaneous Clustering and Feature-Selection via Co-Evolution

Our clustering application involves the optimization of three quantities, which together form a complete solution, (a) the set of features (dimensions) used for clustering; (b) the actual cluster centres; and (c) the total number of clusters. Since this is the case, and since the relationship between the three quantities is complementary (as opposed to adversarial), it makes sense to use cooperative (as opposed to competitive) co-evolution as the model for the overall genetic optimization process.

Insert Figure 10 Here.

In similarity-based clustering techniques, a number of cluster centres are proposed. An input pattern (point) is assigned to the cluster with the centre closest to the point. After all the points are assigned to clusters, the cluster centres are re-computed. Then, the points are re-assigned to the (new) clusters based on their distance from the new cluster centres. This process is iterative, and hence it continues until the locations of the cluster centres stabilize.

During co-evolutionary clustering, the above occurs, but in addition, less discriminatory features are eliminated, leaving a more efficient subset for use. As a result, the overall output of the genetic optimization process is a number of traditionally good (i.e. tight and well-separated) clusters, which also exist in the smallest possible feature space.

The co-evolutionary GA utilizes two populations: one of cluster centres and another of dimension selections. Each population goes through a typical GA process. This process is iterative and follows these steps:

(a) fitness evaluation;

(b) convergence testing (to decide whether to exit or not);
(c) selection;
(d) the application of crossover and mutation (to generate the next population);
(e) back to (a).

This continues until the convergence test is satisfied and the process is stopped.

The GA process is applied to the first population and in parallel (but totally independently) to the second population. The critical difference between a traditional GA and one applied to two or more co-evolving populations is that the fitness of an individual in one population is evaluated after that individual is (temporarily) joined to an individual or individuals in the other population(s), to form a complete candidate solution. Hence, the fitness of individuals in one population is coupled with the fitnesses of individuals in the other population(s).

Below, is a description of the most important aspects of the genetic algorithm applied to the co-evolving populations that make-up PalmPrints. First, the way individuals are represented (as chromosomes) is described. This is followed by an explanation of step (a) to step (d), listed above, but not in that order. Finally, a discussion of the results is presented.

A. Chromosomal Representation

In any co-evolutionary genetic algorithm, two (or more) populations co-evolve. In our case, there are exactly two populations, (a) a population of cluster centres (*Cpop*), each represented by a variable-length vector of real numbers; and (b) a population of 'dimension-selections', or simply dimensions (*Dpop*), each represented by a vector of bits. Each individual in *Cpop* represents a (whole) number of cluster centre coordinates. The total number of coordinates equals the number of clusters. On the other hand, each individual ('dimension-selection') in *Dpop* indicates, via its 1-valued bits, which dimensions (or features) will be used, and which, via its 0-valued bits, will not. Splicing an individual (or chromosome) from *Cpop* with an individual (or chromosome) from *Dpop* will give us a 'complete' chromosome that has the following form:

$\{(F_{11}, F_{21}... F_{n1}), (F_{12}, F_{22}... F_{n2}) ... (F_{1m}, F_{2m}, F_{nm}), 10110000000...0\}$

Taken as a single representational unit, this chromosome determines:

(a) The *number of clusters*, via the number of cluster centres (*m*) in the left-hand side of the chromosome;
(b) The actual *cluster centres*, via the coordinates of cluster centres, also presented in the left-hand side of the chromosome; and
(c) The *number of dimensions*, or features (*n*) used to characterize the cluster centres, via the bit vector on the right-hand side of the chromosome.

As an example, in our experiments, the chromosome allows up to *m* clusters (with *m* set to 26), in a space with up to *n* dimensions (with *n* set to 84). However, the actual number of dimensions, for this particular vector, is 3. This is so because the bit vector, which is *n*-bits long, has 1 in its first bit location, 1 in its third bit location and 1 in its fourth bit location; the other bit locations all have 0's.

B. Crossover and Mutation, Generally

Crossover exchanges information between two parent chromosomes and introduces new information for the child chromosomes. Mutation is the second way a genetic algorithm introduces new information (and hence diversity) into the next offspring. It can introduce new traits not in the original population, and it also prevents the genetic algorithm from converging too fast, or stagnating in sub-optimal local maxima.

In our approach, the crossover operators need to (a) deal with varying-length chromosomes; (b) allow for a varying number of feature dimensions; (c) allow for a varying number of clusters; and (d) be able to adjust the values of the coordinates of the various cluster centres. This is not a trivial task, and it is achieved via a number of crossover operators, each devised for its own task. This is explained below.

C. Crossover and Mutation for Cpop

Cpop requires crossover and mutation operators suited for variable-length chromosomes as well as real-valued parameters**.**

When crossing over two parent chromosomes to produce two new child chromosomes, the algorithm follows a three-step procedure:

1. The length of a child chromosome is randomly selected from the range: [*2*, *MaxLength*], where *MaxLength* is equal to the total number of clusters in both parent chromosomes;
2. Each child chromosome picks up copies of cluster centres, from each of the two parents, in proportion to the relative fitnesses of the parents (to each other). For example, if there were two parent chromosomes, *A* and *B*, with fitnesses 0.9 and 0.3 respectively, then a child chromosome *C* (with a length, determined in step 1, to be equal to 9) would pick up 0.9/(0.9+0.3) * 9 = 6.75, rounded to 7

cluster centres (randomly chosen) from parent *A,* and the rest of the cluster centres - 2 - randomly chosen, from parent *B*; and finally,

3. The actual values of the cluster coordinates are modified using the following (mutation) formula:

$$f_i = min(f_i) + \alpha\,[max(f_i) - min(f_i)] \quad i=1,2,\ldots \quad (13)$$

f_i: feature i
$min(f_i)$: minimum value that feature i can take
$max(f_i)$: maximum value that feature i can take
α: a real number in [0,1]

In addition to crossover, mutation is applied, to a randomly chosen cluster centre, with a relatively high probability (μ_c) of 0.02 (or 2%).

D. Crossover and Mutation for Dpop

Dpop needs one crossover operator suited for fixed length binary-valued parameters. For a binary representation of *Dpop* chromosomes, single-point crossover is applied. Following that, mutation is applied with a mutation rate of (μ_d). of 0.02 (or 2%).

E. Selection and Generation of Future Generations

The creation of future generations is achieved via elitism, various ways of selection, as well as crossover and mutation (explained above).

For both populations, elitism is applied first, and causes copies of the fittest chromosomes to be carried over (without change) from the current generation to the next generation. Elitism is set at 10%. Another 10% is generated via the crossing over of pairs of elite individuals, to generate an equal number of children. The rest (80%) of the next generation is generated through the application of crossover and mutation (in that order) to randomly selected individuals from the non-elite part of the current generation. Crossover is applied with a probability of 1 (i.e. all selected individuals are crossed over), while mutation is applied with a probability of 2%.

F. Fitness Function

The widely used formula for the computation of Mean Square Error (MSE) is chosen as the basis for our fitness function. But, since Mean Square Error (MSE) can always be decreased by adding more cluster centres, a fitness formula that uses MSE is a monotonically decreasing function of the number of clusters. Such a fitness function is poorly suited to clustering applications where the number of clusters is not only unknown, but also variable.

A method that accepts a dynamically-variable number of clusters was developed by Lee [9]. The fitness formula for this method is:

$$MSE \quad heuristic \quad fitness = \sqrt{n+1}\,\frac{\sum_{i=1}^{n}\sum_{j=1}^{m_i} d(c_i, x_j^i)}{n} \quad (14)$$

n: dynamic no. of clusters
m_i: the number of data points belonging to the i_{th} cluster
c_i: the i_{th} cluster centre
x_j^i: the j_{th} data point belonging to the i_{th} cluster
d(a,b): the Euclidean distance between points a and b

The heuristic function penalizes model complexity by multiplying the MSE fitness by a factor proportional to the square root of the number of clusters. The penalization factor was chosen primarily because it provided good clustering results for a variety of data sets.

MSE heuristic fitness, shown above, was applied (in [9]) to dynamic clustering, but with a constant number of dimensions (2). If the number of dimensions is also variable, then we need to amend the formula above to take that fact into account. Our approach does this. It implements dynamic clustering in parallel with feature selection. This leads to the elaboration of the formula above. This extended MSE fitness, especially developed for our GA, measures quantities of both cluster tightness (f_T) and cluster separation (f_S), then uses them in one overall equation for computation of fitness.

$$MSE \quad extended \quad fitness = \sqrt{n+1}\,(f_T + \frac{1}{f_S}) \quad (15)$$

$$f_T = \sum_{i=1}^{n}\sum_{j=1}^{m_i} d(c_i, x_j^i)/n \quad (16)$$

$$f_S = \sqrt{k+1}\sum_{i=1}^{n} d\{c_i, Ave(\sum_{j=1, j\neq i}^{n} c_j)\} \quad (17)$$

n: dynamic no. of clusters
k: dynamic no. of features
c_i: the i_{th} cluster centre
m_i: the number of data points belonging to the i_{th} cluster
x_j^i: the j_{th} data point belonging to the i_{th} cluster
d(a,b): the Euclidean distance between points a and b
Ave(A): the average value of A

The square root of the number of clusters and the square root of the number of dimensions in the MSE

extended fitness formula are chosen to be unbiased in the dynamic co-evolutionary environment. The point of the MSE extended fitness is to optimize fitness by minimizing the within-cluster spread and maximizing the inter-cluster separation.

G. Convergence Testing

The number of generations prior to termination depends on whether an acceptable solution is reached or a pre-set number of iterations is exceeded. Most genetic algorithms keep track of some population statistics such as maximum and mean fitness, standard deviation of (maximum or mean) fitness, and minimum cost. Any of these or any combination of these can serve as a convergence test. In PalmPrints, we stop the GA when the maximum fitness does not change by more than .0001 for 10 consecutive generations.

H. Experimentation Platform

The machine used for running our experiments was a Pentium 3 PC, with 128 mega-bytes of RAM and a 30 mega-byte hard disk drive. Microsoft Visual C++ 6.0 was used for implementing the co-evolutionary clustering algorithm. On average, a full clustering run took 43 minutes to conclude.

V. Implementation Results

Deciding the sizes of the initial and following generations is difficult. There is a trade-off between population size and the total number of generations to convergence. Experience indicates that the most effective population size is dependent on the problem being solved, the representation used, and the operators manipulating the representation [17].

Insert Figure 11 Here.

Insert Figure 12 Here.

The experiment presented here used 1000 images (5 x 200 hands) and 84 normalized features. Fitness convergence results are shown in fig. 11. Despite the complexity of the multi-optimization task, convergence occurred at, or before, the 250th generation. (As mentioned, convergence occurs when maximum fitness does not change by more than 0.0001 for 10 consecutive generations.) The results are clear and promising. The whole experiment was repeated 200 times. This produced an *average* co-evolutionary clustering fitness of 0.973 (with a significantly low standard deviation of 0.111). The *best* clustering fitness achieved is 0.991. The average number of *clusters* is a reasonable 4 (with a low standard deviation of 0.471) with varying number of images in each (57,14,14 and 15). Each hand has five images. If the majority (say 4) of those images fell into a given cluster, while the fifth image went into a different cluster, then we would say that that 5^{th} image was misplaced. Overall misplacement rate is computed by finding the total number of individual images that were misplaced, and then dividing that number by the total number of images used in a given trial. In our experiments, average hand image *misplacement rate* is 0.058.

Following convergence, the *dimension* of the feature space comes down to 41. Hence, the algorithm considerably reduces the complexity of the problem, by de-selecting more than half the number (84) of features initially available to the algorithm. It is worth noting that the *average number of features* (shown in fig. 12) starts at 40.02, and not 84. This is because this number represents the average number of (selected) features in a Dpop chromosome. Each Dpop chromosome is made of a string of 84 bits. During initialization of the very first Dpop generation, and for every chromosome in that generation: each one of a chromosome's 84 bits is *randomly* assigned a value of either 1 or 0 (selected or not). Hence, on average, half the bits in a Dpop chromosome end up with 1's and the other half with 0's. Half of 84 is 42, which is almost equal to the empirically observed value of 40.02. Towards the end of the run, the number (and type) of features converges and stabilizes at 41. However, this 41 represents the same set of 41 dimensions or features, and not different sets of 40 or so features, which was the case at the start of the run.

Insert Figure 13 Here.

Those 41 features are partitioned into four groups as shown in fig. 13.

VI. Conclusions & Future Work

This study is the first to use a Genetic Algorithm to simultaneously achieve dimensionality reduction and object (hand image) clustering. In order to do this, a cooperative co-evolutionary GA was crafted, one that uses two populations of part-solutions in order to evolve complete highly fit solutions to the whole problem. It succeeded in meeting both of its objectives. The results show that the dimensionality of the clustering space is reduced by more than half: from 84 original features to 41 features. Simultaneously, the best fitness achieved is an almost perfect 0.991. Even on average, the number (4) and quality of clusters evolved (given a misplacement rate of 0.058) are quite good. These

results open the way towards other cooperative co-evolutionary applications; in which 3 or *more* populations are used to co-evolve solutions and designs consisting of 3 or *more* loosely coupled sub-solutions or modules.

In addition to the main contribution of this study, it also offers a number of other significant results, and tools for further research in Biometrics as well as GAs.

First, empirical results (shown in Table III) demonstrate that reorienting the images of the fingers of a hand prior to any feature extraction consistently leads to more *stable* feature values, *regardless* of the features measured.

Second, the authors introduced a number of new or modified structural (e.g. palm aspect ratio) and statistical *features* (e.g. finger 1D contour transformation) that were used in our clustering algorithm, and may prove equally useful to others working in Biometric-based technologies.

Third, there is also a small 1000 (right-) hand image CD *database* that includes 5 greyscale images of the right hand of each of 200 different people, in addition to the contour of these images, represented in two formats: image and coordinates of contour.

As for the future, one not so trivial challenge to the speedy proliferation of co-evolutionary (multi-population) techniques lie in the large amounts of computing power necessary (and not easily obtainable) for such applications. There is also the problem of manual fine-tuning of many GA and simulation parameters, one that seems to render the autonomy of GA methods somewhat hollow. In line with the above, one can say that the results of our particular study can be improved via (a) further fine-tuning of the GA parameters – a self-tuning parameter-less GA is currently under development; and (b) use of more powerful computers that would allow us to run larger populations for longer – a Beowulf cluster has been constructed and will be used for implementing a parallelized version of the algorithm.

ACHNOWLEDGMENTS

The authors wish to thank F. Cianci, H. Patel, P. Prakash, and E. Tochtmais for their support in collecting the hand image database, as well as devising and implementing most of the pre-processing algorithms. Any researcher interested in receiving a copy of the hand images database (which includes both raw and pre-processed images) is encouraged to contact Dr. N. Kharma directly at: kharma@ece.concordia.ca. I also wish to thank Mr. Ashraf Nijim for his help in type-setting the final version of the paper.

REFERENCES

[1] A. K. Jain, A. Ross and S Pankanti, "A prototype hand geometry-based verification system," in *2nd Int'l Conference on Audio- and Video- based Biometric Person Authentication*, Washington D.C., 1999, pp. 2-7

[2] J. You, W. Li and D. Zhang, "Hierarchical palmprint identification via multiple feature extraction," *Pattern Recognition* 35, pp. 847-859, 2002.

[3] I.-S.Oh, J.-S Lee and B.-R Moon, "Local search-embedded genetic algorithms for feature selection," *Proc. of International Conference. on Pattern Recognition*, 2002, pp. 148-151.

[4] M. L. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn, and A. K. Jain, "Dimensionality reduction using genetic algorithms," *IEEE Transaction on Evolutionary Computation,* Vol. 4, No.2, pp. 164-171, July 2000.

[5] W.L.G. Koontz, P.M. Narendra, K. Fukunaga, "A branch and bound clustering algorithm," *IEEE Transaction Comput.* C-24, pp. 908-915, 1975.

[6] J.H. Wolfe, , "Pattern clustering by multivariate mixture analysis," *Multivariate Behav. Res.* 5, pp. 329-350, 1970.

[7] W.L.G. Koontz, P.M. Narendra, K. Fukunaga, "A graph theoretic approach to nonparametic cluster analysis," *IEEE Transaction Comput.* C-25, pp. 936-944, 1975.

[8] C.W. Therrien, "Decision estimation and classification: and introduction to pattern recognition and related topics," John Wiley & Sons, Inc. Press, 1989, pp. 217-218.

[9] C.-Y. Lee, "Efficient automatic engineering design synthesis via evolutionary exploration," Ph.D. dissertation, Dept. Mechanical Eng., California Institute of Technology, Pasadena, California, 2002.

[10] L. Y. Tseng, S. B. Yang, "A genetic approach to the automatic clustering problem," *Pattern Recognition* 34, pp. 415-424, 2001.

[11] U. Maulik, S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *Pattern Recognition* 33, pp. 1455-1465 2000.

[12] C. A. Pena-Reyes, M. Sipper, "Fuzzy CoCo:A cooperative-coevolutionary approach to fuzzy modeling," *IEEE Transaction on Fuzzy Systems* Vol. 9, No.5, pp. 727-737, October 2001 .

[13] Windows 20/20 from byLight Technologies, Available: http://www.hotfreeware.com/.

[14] Mineichi Kudo, Jack Sklansky, "Comparison of algorithms that select features for pattern classifiers," *Pattern Recognition* 33, pp. 25-41, 2000.

[15] G.Keller and B.Warrack, "Statistics for management and economics," 5th ed., Duxbury Press, 2000, pp.75-76.

[16] W. Wang, Z. Bao, Q. Meng, G.M. Flachs, J.B.Jordan and J. J. Carlson, "Hand recognition by wavelet transform and neural network," *SPIE,* vol. 2484, pp. 347-353, 1998.

[17] Randy L. Haupt and Sue Ellen Haupt, "Practical genetic algorithms," in Plastics, 1st ed. vol. 1, Wiley Interscience, New York, 1998, pp. 91-92.

[18] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *Pattern Recognition* 33 (2000) 1455-1465.

[19] R. Cucchiara, "Genetic algorithms for clustering in machine vision," *Machine vision and application* 11:1-6, 1998.

TABLE I
PROBLEMS OF CONTOUR OUTPUT

	Total 1000 images								
Problems	A	B	C	D	E	F	G	H	Others
Numbers	6	79	6	3	5	11	15	42	5

TABLE II
ROTATION ANGLES

	Finger Rotation Angles				
Fingers	Little finger	Ring finger	Middle finger	Second finger	The thumb
Degrees	31.47°	10.35°	-0.674°	-22.07°	-37.39°

TABLE III
FEATURE MEASUREMENT BEFORE & AFTER FINGER REORIENTATION

	Finger Before Reorientation					
CENTRAL	S1	S2	S3	S4	S5	CV(%)
M1	0.4432	0.4322	0.4450	0.4382	0.4394	1.01
M2	0.2194	0.2140	0.2205	0.2171	0.2175	1.01
M3	0.1733	0.1679	0.1743	0.1705	0.1712	1.31
M4	0.1655	0.1583	0.1669	0.1612	0.1626	1.87
FOURIER	S1	S2	S3	S4	S5	CV(%)
A1	0.2768	0.2761	0.2772	0.2786	0.2776	0.291
A2	0.1326	0.1310	0.1329	0.1324	0.1323	0.510
A3	0.0883	0.0873	0.0885	0.0882	0.0881	0.490
A4	0.0662	0.0654	0.0664	0.0661	0.0661	0.496
A5	0.0531	0.0524	0.0532	0.0530	0.0529	0.500
ZERNIKE	S1	S2	S3	S4	S5	CV(%)
Z20	0.2975	0.2903	0.2956	0.2886	0.2879	1.33
Z22	0.0930	0.0901	0.0921	0.0891	0.0888	1.82
Z31	0.2227	0.2162	0.2202	0.2140	0.2130	1.69
Z33	0.0567	0.5479	0.0560	0.0541	0.0538	2.04
Z42	0.1053	0.1019	0.1037	0.1006	0.1000	1.93
Z44	0.0222	0.0214	0.0219	0.0211	0.0209	2.25
GEOMETRIC	S1	S2	S3	S4	S5	CV(%)
Length	122.7	122.1	122.7	120.5	120.8	0.766
Midline	119.4	119.1	119.2	118.5	118.7	0.282
Tip angle	15.19	15.26	15.67	15.95	16.40	2.860
Height	119.2	119.0	119.2	118.5	118.5	0.261
	Finger After Reorientation					
CENTRAL	S1	S2	S3	S4	S5	CV(%)
M1	0.3079	0.3091	0.3089	0.3105	0.3106	0.335
M2	0.1485	0.1486	0.1485	0.1492	0.1492	0.224
M3	0.1165	0.1170	0.1169	0.1176	0.1177	0.383
M4	0.1118	0.1134	0.1134	0.1143	0.1146	0.872
FOURIER	S1	S2	S3	S4	S5	CV(%)
A1	0.2100	0.2094	0.2091	0.2097	0.2095	0.145
A2	0.1029	0.1028	0.1027	0.1030	0.1029	0.113
A3	0.0681	0.0679	0.0679	0.0681	0.0680	0.124
A4	0.0509	0.0508	0.0508	0.0510	0.0509	0.140
A5	0.0405	0.0406	0.0404	0.0406	0.0406	0.144
ZERNIKE	S1	S2	S3	S4	S5	CV(%)
Z20	0.3536	0.3507	0.3500	0.3499	0.3493	0.435
Z22	0.0924	0.0913	0.0910	0.0910	0.0907	0.642
Z31	0.2339	0.2310	0.2301	0.2300	0.2295	0.674
Z33	0.0520	0.0512	0.0510	0.0510	0.0508	0.781
Z42	0.1016	0.1000	0.0995	0.0996	0.0992	0.835
Z44	0.0190	0.0187	0.0186	0.0186	0.0185	0.900
GEOMETRIC	S1	S2	S3	S4	S5	CV(%)
Length	123.4	123.7	124.1	123.9	123.9	0.186
Midline	120.9	120.8	121.1	120.9	120.6	0.141
Tip angle	15.11	15.07	15.50	15.57	16.05	2.320
Height	119.4	119.5	119.5	119.7	119.6	0.081

Si: the ith Sample of a hand.
Mk: the kth order central moments.
Ak: the kth harmonic amplitude of Fourier descriptors.
Znm: the (n+m)th order Zernike moments
Length: inter-finger length.
Midline: finger midline.
Height: finger height.
Tip angle: finger tip angle.
CV(%): equal to the standard deviation of the measurements divided by their mean for samples.

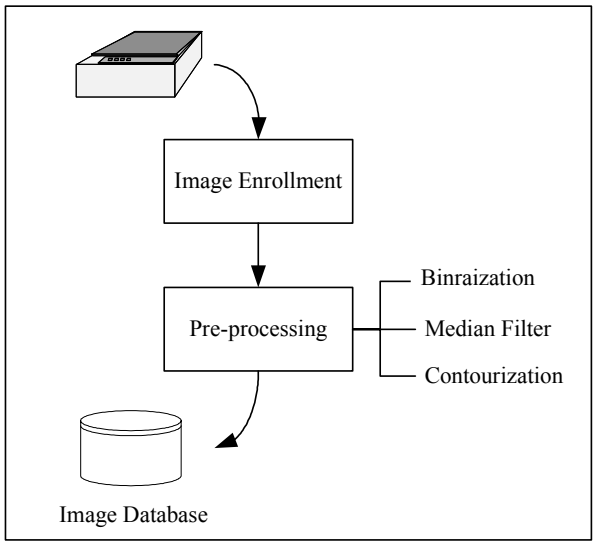

Figure 1: DATA ACQUISITION AND PRE-PROCESSING

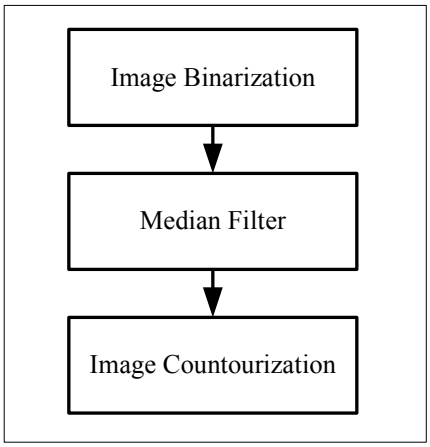

Figure 2: PRE-PROCESSING PROCEDURE

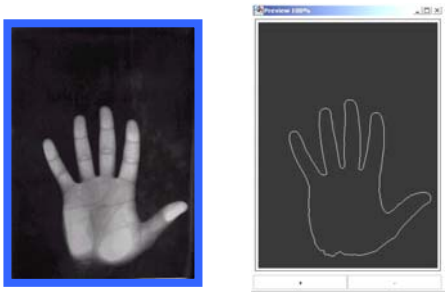

Figure 3: IMAGE PRE-PROCESSING: BEFORE (LEFT) & AFTER (RIGHT)

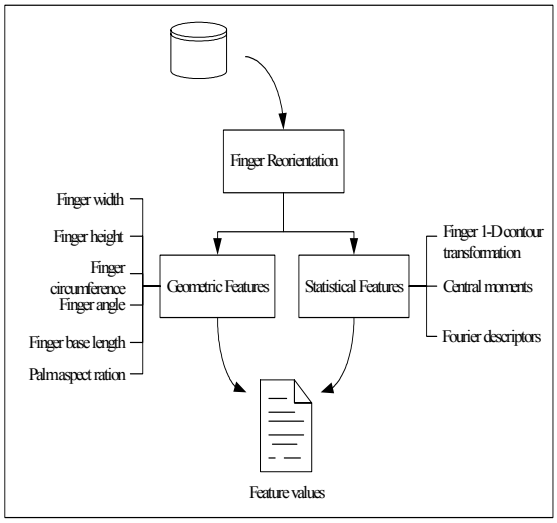

Figure 4: FEATURE EXTRACTION

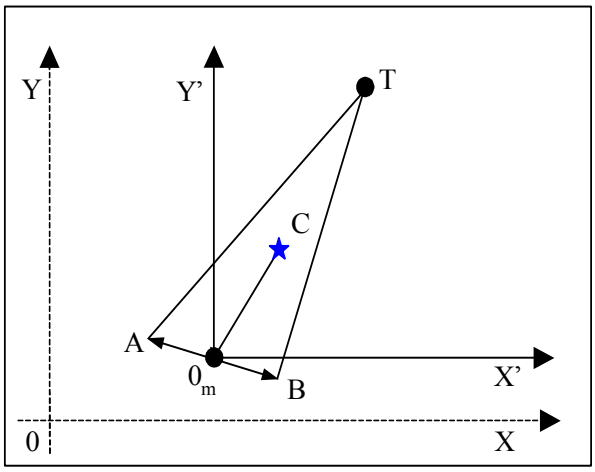

Figure 5: FINGER RE-ORIENTATION

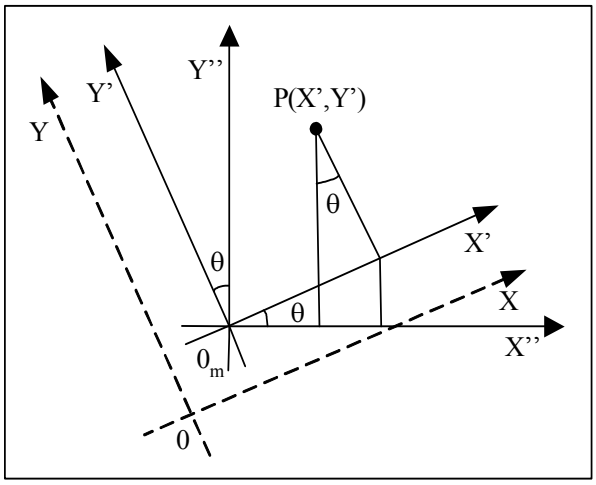

Figure 6: ROTATION AND SHIFTING
OF THE SYSTEM OF COORDINATES

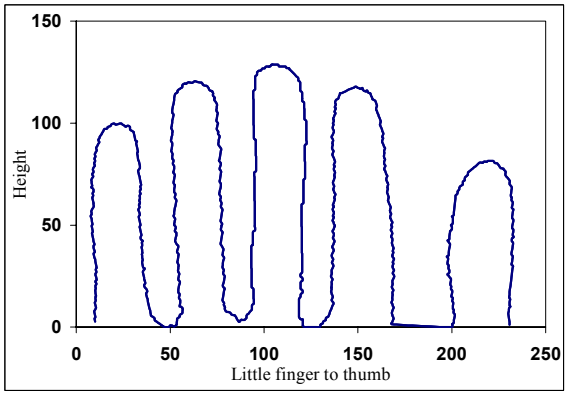

Figure 7: FINGER RE-ORIENTATION

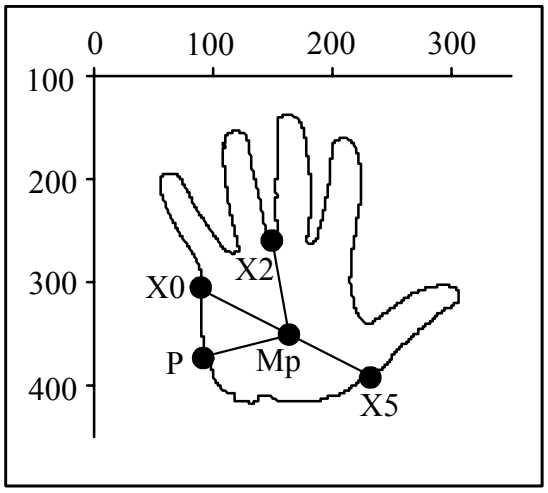

Figure 8: SOME HAND FEATURES

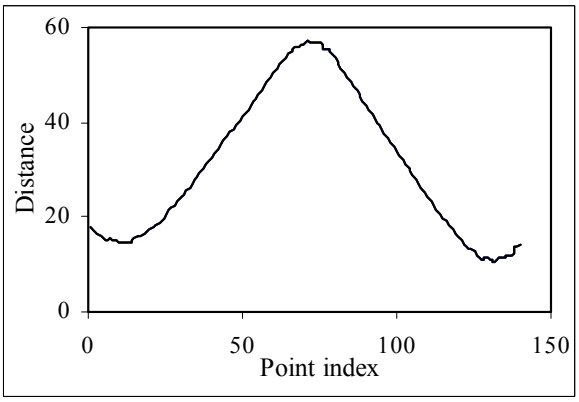

Figure 9: 1D CONTOUR OF A FINGER

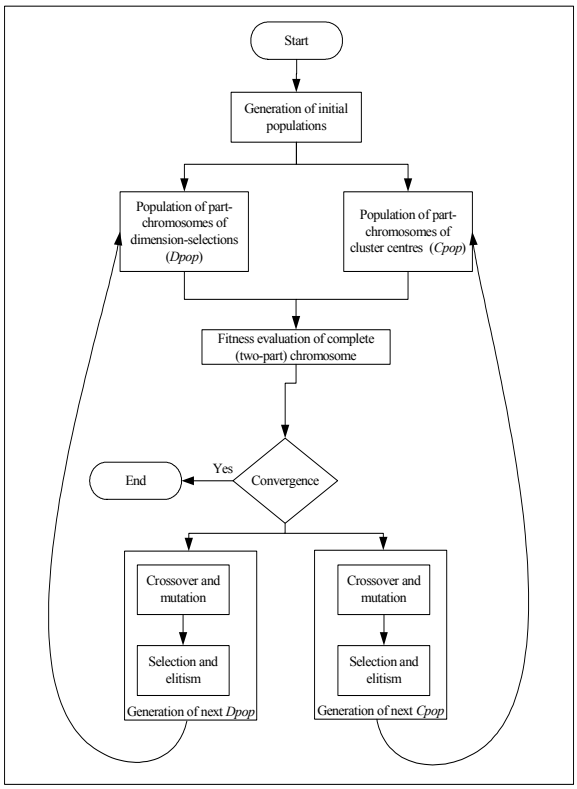

FIGURE 10: CO-EVOLUTIONARY ALGORITHM

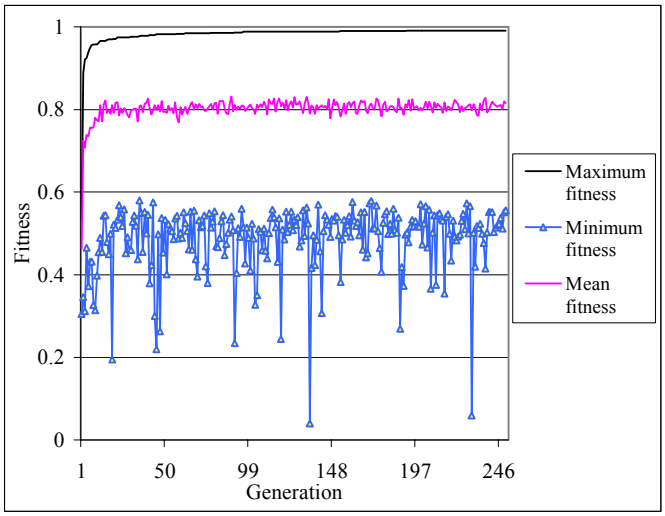

FIGURE 11: FITNESS CONVERGENCE

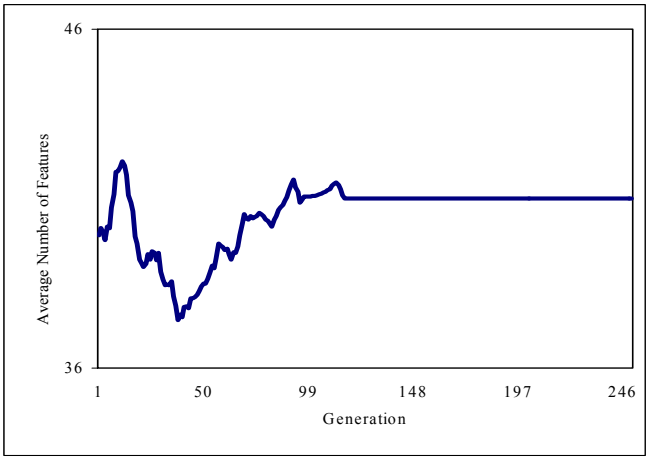

FIGURE 12: FEATURE NUMBER CONVERGENCE

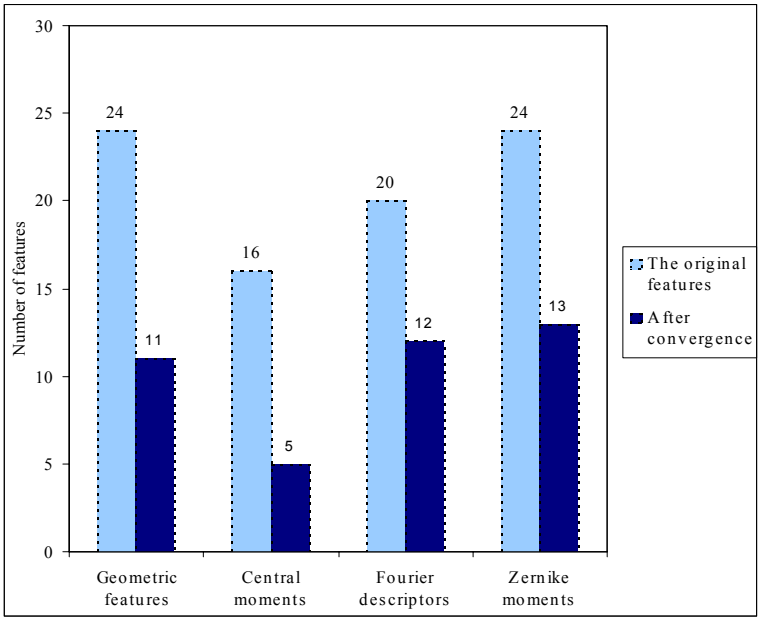

FIGURE 13: FEATURE REDUCTION BY FEATURE TYPE