PalmPrints: A Cooperative Co-Evolutionary Clustering Algorithm For Hand-based Biometric Identification

Pei Fang Guo

A Thesis

In

The Department

Of

Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements
For the Degree of Master of Applied Science at
Concordia University
Montréal, Québec, Canada

June 2003

Your file Votre référence
ISBN: 0-612-83865-X
Our file Notre référence
ISBN: 0-612-83865-X

ABSTRACT

PalmPrints: A Cooperative Co-Evolutionary Clustering Algorithm For Hand-based Biometric Identification

Pei Fang Guo

The thesis first introduces a new adaptive technique of finger upright reorientation by using *the Principle of Coordinate System Rotation.* The empirical results demonstrate that reorienting the images of fingers of a hand prior to any feature extraction consistently leads to more stable feature values, regardless of the features measured. Hand shape analysis included Central Moments, Fourier Descriptors and Zernike Moments is characterized based on 1-D contour transformation.

The main contribution of the thesis is the first to use a genetic algorithm to simultaneously achieve dimensionality reduction and object (hand image) clustering. A novel Cooperative Coevolutionary Clustering Algorithm (CCCA) with dynamic clustering and feature selection has been developed to search for a proper number (without *prior* knowledge of it) of clusters of hand images into these clusters based on a smaller set of new features. In addition to the main contribution of the study, an *MSE Extended Fitness Function* is presented which is particularly suited to an integrated dynamic clustering space.

The proposed design and experimental implementation show that the dimensionality of the clustering space can be cut in half, and the GA evolves an average of 4 clusters with a very low standard deviation of 0.4714. Average hand image misplacement number is 5.8 out of 100 hand images. These results open a new 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.

ACKNOWLEDGEMENTS

First, I would like to express my gratitude and respect to both of my supervisors Dr. Nawwaf Kharma and Dr. Ching Y. Suen, for their constant guidance and constructive suggestions through the entire work of the thesis. They have contributed their ideas, time and effort to enable me to demonstrate this technique by mathematical computation and computer simulation. I really appreciate the opportunity having worked with them.

I am indebted to Francesco Cianci, Hemanshu Patel, Prakash Prakash and Elias Tochtamis who built the database of hand images and made the door open for me to get reach this project. I believe their work is the fundamental basis for my thesis achievement.

Special thanks go to my family for their patience and understanding. They are of always being there with me through the period of my Master study. I owe them every success of this work.

CONTENTS

List of Figures

List of Tables

Chapter 1

Introduction & Review

1.1 Hand Features

1.1.1 Human Hand and Hand Geometry

Anthropologists suggest that humankind survived and evolved due to our large brains and opposing thumbs. The versatile human hand allows us to grasp, throw, and make tools. The physical dimensions of a human hand contain information that is capable of authentication the identity of an individual. This information has been popularly known as hand or palm geometry.

Hand geometry, as the name suggests, refers to the geometric structure of the hand. General speaking, hand geometry is an easy biometric methodology to deploy, is likely to elicit few user-related objections and will usually perform well in most situations. Against this is the fact that the readers are a little bulky and are not easy to blend seamlessly into other equipment. Hand geometry provides for a good general purpose biometric, with acceptable performance characteristics and relative ease of deployment coupled to a low learning curve for users. It offers good flexibility in its configuration, catering for stand-alone or networked use.

1.1.2 Measurable Features of the Hand

There exists a wealth of information within the geometry of an individual hand. Since each human hand is unique, finger length, width, thickness, curvatures and relative location of these features distinguish one human being from others. Hand geometry works by taking a three dimensional view of the hand in order to determine the geometry and metrics around these finger length, height and other details.

Palm is the inner surface of a hand between the wrist and the fingers. There are many features exhibited in a palm. These features are: aspect ratio of palm; palm size and form of the lines on the palm; the hollowness of the palm. Comparing with the palm shape features, the relatively stable feature extracted from the hands is the print of palm. In particular, the lines on a person's hand are unique to every individual; even our own two hands are never quite alike. For example, there are three principal lines caused by flexing hand and wrist, which are named as heart line, head line and life line, respectively[YLZ02]. The location and form of these principal lines in a palm are very important physical features for identifying an individual because they vary slowly from time to time. Due to the stability of these feature lines, they can be regarded as reliable and stable features to distinguish a person from others. Besides, there are numerous attributes which make the palm lines on a person's hand distinguishable from the lines on another. These include: color, clarity, length, position within the palm, continuity and variations in thickness.

In addition to hand geometry features, the methodology of shape analysis, a conventional method for object identification, has been extensively employed in the field of object recognition by various researchers on different shapes. The idea behind the chosen analysis is to consider human hand as an object of definite rigid shape, extract its boundary and shape parameters as recognition features, often normalized and invariant to various geometric change such as translation, rotation and scale. Shape analysis includes geometric moments, invariant geometric moments, Zernike moments, Fourier descriptors and wavelet analysis.

1.1.3 Review of Features used in Previous Studies

Research on hand features has drawn considerable attention over the last 25 years [JRP99][YLZ02]. The paper [JPR98] addressed the issue of biometric-based access to the Web. The feasibility of such a system was demonstrated by designing a prototype system which used 16 hand geometric features, such as the lengths and widths of the fingers, aspect ratio of the palm or fingers, thickness of the hand, to verify a user and give him access to particular files on the Web. It developed a prototype system which used hand geometry to authenticate users to restrict access to specific web pages.

A novel approach was presented to authenticate individuals by using their palm-print features[HCL03]. The palm features are extracted from the ROL, a square region in a palm which is called region of interest, by using Sobel and morphological operations. In extracting the features, it used the operator-based approach to extract the line-like features of palm-print in the ROI of palm table. The ROI images were uniformly divided into several small grids. The mean values of pixels in the grids were calculated to obtain the feature values. These values were sequentially arranged row by row to form the feature vectors.

1.2 Pattern Classification vs. Clustering

1.2.1 Definitions of Classification and Clustering

The goal of pattern recognition is to classify objects of interest into a known number of categories or *classes*. Pattern classification is the process of inferring a class given a set of observations. The set of processes or events to be classified could be a set of physical objects or a set of more abstract mental states. The processes or events with some similar properties are grouped into a class. The total number of pattern classes in a particular problem is often determined by the particular application in mind. For example, consider the problem of English character recognition; we should have a problem of 26 classes (ignore their upper or lower case). On the other hand, if we are interested in discriminating English characters from Chinese characters, we have only a two-class problem.

Classification definition:

The concept of classification may be expressed in terms of the partition of feature space (or a mapping from feature space to decision space). Suppose that N features are to be measured from each input pattern. Each set of N features can be considered as a vector, X, called a feature (measurement) vector, or a point in the N-dimensional feature space, Ω_x. The problem of classification is to assign each possible vector or point in the feature space to a proper pattern class. This can be interpreted as a partition of the feature space into mutually exclusive regions and each region will correspond to a particular pattern class. Mathematically, the problem of classification can be formulated in terms of "discriminant function". Let $\omega_1, \omega_2, \ldots, \omega_m$ be designated as the m possible pattern classes to be recognized, and let $X = [x_1, x_2, \ldots, x_N]$ be the feature (measurement) vector where x_i represents the ith feature measurement. Then the discriminant function $D_j(X)$ associated with pattern class ω_j, $j = 1, \ldots, m$, is such that if the input pattern represented by the feature vector X is in class ω_j, denoted as $X \sim \omega_j$, the value of $D_i(X)$ must be the largest, i.e., for all $X \sim \omega_j$,

$$D_i(X) > D_j(X), \quad i, j = 1, \ldots, m, \quad i \neq j$$

$$\text{with} \quad D_i(X) - D_j(X) = 0$$

If there exists some set of patterns, the individual classes of which are already known, then one has a problem in *supervised pattern recognition*. If the classes of all of the available patterns are unknown, and perhaps even the number of these classes is unknown, then one has a problem in *unsupervised pattern recognition* or *clustering*. Clustering is the grouping of similar objects. In clustering problems, one attempts to find classes of patterns with similar properties where sometimes even these properties may be undefined.

Clustering is the process of counting and labeling of objects within an image. Clusters consist of point groupings that are related to one another by a predetermined measure. This measure can be defined as a distance between clusters or a similarity measure such as point value, or it may be a complex set of identifiers and constraints that define membership in a cluster.

Clustering definition:

In partitional clustering we are interested in generating a single partition –to describe the groups or clusters present in the data. A formal definition of partitional clustering can be described as follows. Given a collection of n pattern vectors where each pattern X is a *m*-dimensional vector characterized by the set of features ($x_1, x_2, \ldots, x_m$), find the clusters present in the data. A cluster is defined by the similarity of patterns present in it. The number of clusters may be known or unknown.

1.2.2 Commonly used Classification Techniques

Non parametric classification refers to a set of procedures for decision making that can be used when there is only partial information about the class conditional density functions. For example, one may know that a certain parameter of the density such as the mean or variance lies only within a range of possible values. Alternatively, one may know nothing about the density function except that it has a positive mean and is symmetric as it seeks to develop tests that do not depend on the unknown information. Several basic classification methods are discussed below.

A. Nearest-neighbour classification (Piecewise linear discriminant functions)

Nearest-neighbour methods provide an important data classification tool for recognising object classes in pattern recognition domains. Nearest-neighbour methods have been used as an important pattern recognition tool. In such methods, the aim is to find the nearest-neighbour of an unidentified test pattern within a hyper-sphere of pre-defined radius in order to determine its true class. The traditional nearest-neighbour rule has been described as follows:

- Out of N training vectors, identify the k nearest neighbours, irrespective of class label. *k* is chosen to be odd.
- Out of these *k* samples, identify the number of vectors, k_i, that belong to class ω_i, i = 1, 2,..., M. Obviously $\sum_i k_i = k$.
- Assign *x* to the class ω_i with the maximum number k_i of samples.

Nearest-neighbour methods can detect a single or multiple number of nearest neighbours. A single nearest-neighbour method is primarily suited to recognising data where we have sufficient confidence in the fact that class distributions are non-overlapping and the features used are discriminatory.

In *k* nearest-neighbour methods, certain implicit assumptions about data are made in order to achieve a good recognition performance. The first assumption requires that individual feature vectors for various classes are discriminatory. This assumes that feature vectors are statistically different across various classes. This ensures that for a given test is more likely to be surrounded by data of its true class rather than of different classes. The second assumption requires that the unique characteristic of a pattern that defines its signature, and ultimately its class, is not significantly dependent on the interaction between various features.

In other words, nearest-neighbour methods work better with data where features are statistically independent. This is because nearest-neighbour methods are based on some form of distance measure and nearest-neighbour detection of test data is not dependent on their feature interaction.

B. Linear discriminant classification

In this case, a linear combination of the feature measurements $(x_1, x_2, \ldots, x_N)$ is selected for Di(X), i.e.,

$$D_i(X) = \sum_{k=1}^{N} w_{ik} x_k + w_{i,N+1} \qquad \text{i} = 1, \ldots, m$$

The decision boundary between regions in Ω_x associated with ω_i and ω_j is in the form of

$$D_i(X) - D_j(X) = \sum_{k=1}^{N} w_k x_k + w_{N+1} = 0$$

with $$w_k = w_{ik} - w_{jk} \quad \text{and} \quad w_{N+1} = w_{i,N+1} - w_{j,N+1}$$

In practice, the proper values of the coefficients or weights $w_1, w_2, \ldots, w_{N+1}$ are usually not available. Under such a circumstance, it is proposed that the classifier is designed to have the capability of estimating the best values of the weights from the input patterns. The basic idea is that, by observing patterns with known classification, the classifier can automatically adjust the weights in order to achieve correct recognition. The performance of the classifier is supposed to improve as more and more patterns are applied. This process is called training or learning, and the patterns used as the inputs are called training patterns.

C. Minimum distance classification

An important class of linear classifiers is that of using the distances between the input pattern and a set of reference vectors or prototype points in the feature space as the classification criterion. Suppose that m reference vectors $R_1, R_2, \ldots, R_m$ are given with R_j

associated with the pattern class ω_j. A minimum-distance classification scheme with respect to R1, R2, ..., Rm is to classify the input X as from class ω_i, i.e.,

$$X \sim \omega_j \text{ if } | X - R_i | \text{ is the minimum}$$

Where | X – Ri | is distance defined between X and Ri.

1.2.3 General Clustering Techniques

An alternative, and in some sense equivalent, method for defining the classes is to seek clusters of the points in the measurement space. The procedure is known as clustering. The method for finding the clusters may have a heuristic basis or may be more rigorously dependent on minimization of a mathematical clustering criterion. In either case, iterative procedures are generally used to find the clusters.

A. k-means algorithm

The simplest and the most well known clustering algorithm is the k-means algorithm, which groups *N* samples, corresponding labels with ω_1 or $\omega 2$ or ... ω_{N_c}, into *Nc* clusters by choosing a similarity measure that is the Euclidean distance of the samples and a criterion *J* that measures the optimality of the classification. The criterion function *J* is defined by:

$$J = \sum_{k=1}^{N_c} \sum_{y^{(i)} \sim \omega_k} | y^{(i)} - \mu_k |^2$$

where the second sum is over all samples in the *k*th cluster and μ_k is the "center" of the cluster to be the sample mean of the *k*th cluster. When μ_k is fixed, *J* is minimized by choosing the class of $y^{(i)}$ as the class of the cluster with the nearest mean. However, this algorithm requires that the user specify the desired number of clusters.

B. Minimum square-error algorithm

The minimum squared-error is a well known criterion used to obtain a specified number of clusters. The squared-error for a set of *n* m-dimensional patterns is given by

$$E_k^2 = \sum_{k=1}^{K} e_k^2$$

where K is the desired number of clusters, and e_k^2 is defined as follows:

$$e_k^2 = \sum_{i=1}^{n_k} (x_i^{(k)} - m^{(k)})^t (x_i^{(k)} - m^{(k)})$$

where

$$m^{(k)} = \frac{1}{n_k} \sum_{i=1}^{n_k} x_i^{(k)}, \qquad n = \sum_{k=1}^{K} n_k$$

C. The ISODATA algorithm

A well-known ISODATA clustering algorithm provides a means for determining the number of clusters *N*c. In particular, it includes heuristic provisions for splitting an existing cluster into two and for merging two existing clusters into a single cluster.

The main procedure for ISODATA is always try to split if the number of clusters is less than half the desired number, to always try to merge if the number of clusters is more than twice the desired number, and otherwise to try to merge or to split on every other iteration. Note that *N*c, the number of clusters, can be changed by each subprocedure.

ISODATA assumes that the user enters with an estimated or desired number of clusters. Call this number N_D. Then the following quantities are computed for each of the existing clusters

$$d_k = \frac{1}{N_k} \sum_{y^{(i)} \sim \omega_k} | y^{(i)} - \mu_k | \qquad k = 1, 2, \ldots, N_c$$

$$\sigma_k^2 = \max_j \frac{1}{N_k} \sum_{y^{(i)} \sim \omega_k} (y_j^{(i)} - \mu_{k_j})^2 \qquad k = 1, 2, \ldots, N_c$$

where N_k is the number of samples in cluster k, the sum is over all samples in the cluster, and the subscript j refers to the jth component of the vector. The quantity d_k is the average distance of samples from the mean for the kth cluster and the σ_k^2 is the largest variance along the coordinate axes. The overall average distance of samples d is defined by

$$d = \frac{1}{N} \sum_{k=1}^{N_c} N_k d_k$$

A maximum spread parameter σ_s^2 is also assumed to have been input by the user. Now, if for any k, σ_k^2 is greater than σ_s^2, then the cluster is split provided that either

$$D_k > d \quad \text{and} \quad N_k > 2(T + 1)$$

Or the number of clusters N_c is less than $N_D/2$. T is the threshold on number of samples in a cluster.

The ISODATA method for splitting is to replace the original cluster center by two centers displaced slightly in opposite directions along the axis k of largest variance. Usually, an amount of displacement equal to a small fraction of σ_k is satisfactory.

D. Vector quantization algorithm

Vector quantization can be significantly more efficient for signal coding than ordinary scalar quantization. That is, fewer bits are generally required if a block of data is coded as a vector than if the samples are quantized and coded individually. The issue of how to choose the center is a problem in clustering.

Vector quantization coding used many ideas that are familiar to pattern recognition but with different terminology. As in pattern recognition a distance function d must be chosen. The cells are defined implicitly by assigning a vector to the center to which it is nearest. That is, vector quantization consists of the mapping

$$y \rightarrow \mu_k$$

where

$$d(y, \mu_k) = \min_j d(y, \mu_j)$$

The centers are selected to minimize a criterion, which is called the *average distortion*. The general form is

$$D = \frac{1}{N} \sum_{k=1}^{N_c} \sum_{y(i) \rightarrow \mu_k} d(y^{(i)}, \mu_k)$$

where N is the total number of vectors $y^{(i)}$. The set of centers is called the "codebook" and the problem of choosing the centers is called "codebook design". When the distance function (also called the distortion function) is taken to be the Euclidean distance, then a suitable codebook design algorithm is the *k*-means algorithm. As a result, the *k*-means algorithm, which is the one most used in practice, is usually referred to in the literature of vector quantization as the Linde-Buzo-Gray (LBG) algorithm [The89].

1.2.4 Classification of Hand Images: Problem Statement

In recognition problems, the patterns are often quite large which makes the idea of describing a complex pattern in terms of a (hierarchical) composition of simple subpatterns very attractive. Also, when the patterns are complex and the number of possible descriptions is very large, it is impractical to regard each description as defining a class (e.g., in hand and face identification problems, recognition of continuous speech, Chinese characters, etc.). Consequently, the requirement of recognition can only be satisfied by a description of each pattern rather than the simple task of classification.

The classification of hand images is based on a set of selected features extracted from hand images. These feature measurements are supposed to be invariant or less sensitive to the commonly encountered variations and distortions and also contain less redundancies. Usually, the decision of what to measure is rather subjective and also dependent on the practical situations. The criterion of feature selection or ordering is often based on either the importance of the features in characterizing the hand patterns or the contribution of the features to the performance of recognition (i.e., the accuracy of hand recognition).

Hand image classification is to compare the feature measures of a hand to known criteria and determine if this hand belongs to a particular class of hands. Although the coarse classification cannot identify a hand uniquely, it is helpful to determine if two different hands can match together.

1.2.5 Review of Previous work in Hand Image Classification

Hand is regarded as one of the most unique, reliable and stable personal characteristics and hand verification provides a powerful means to authenticate individuals for many security systems. Hand feature extraction and matching are two key issues in hand classification.

A work in defining and implementing a biometric system based on hand geometry identification was presented [SSG00] with the 25 initially selected features which included finger widths, finger heights, deviations and angles between the interfinger points and the horizontal. It applied different pattern recognition techniques in classification from Euclidean distance to neural networks. The experimental results were given in False Acceptance Rate (FAR) and False Rejection Rate (FRR). With up to a 97 percent rate of success in classification, the system showed the possibility of using this system in medium/high security environments with full acceptance from all users.

In [SRB01], palmprint classification method based on the orientation property of the palm lines was discussed to distinguish six typical classes. Both principal and coarse wrinkles were defined as line features. Principal lines included heart line, head line and life line respectively detected by using the directional projection algorithm. The palmprint coarse-level classification was analyzed by the number and the corresponding location of the principal lines. Various palmprint images were tested to illustrate the effectiveness of the proposed methods which showed around 97% correct rate of classification could be obtained.

1.3 Genetic Algorithms

1.3.1 Definitions of a Simple GA

One type of natural methods is the genetic algorithm. It is a subset of evolutionary algorithms that model biological processes to optimize highly complex fitness functions. A genetic algorithm allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the "fitness" (i.e., minimizes the cost function).

The basic mechanism in GAs is Darwinian evolution: bad traits are eliminated from the population because they appear in individuals which do not survive the process of selection. The good traits survive and are mixed by recombination (mating) to form better individuals. Mutation also exists in GAs, but it is considered a secondary operator. Its fitness function is to ensure that diversity is not lost in the population, so the GA can continue to explore. The notion of "good" traits is formalized with the concept of building blocks (BBs), which are string templates (schemata) that match a short operation of the individuals and act as a unit to influence the fitness of individuals. The prevailing theory suggests that GAs work by propagating BBs using selection and crossover.

The method was developed by John Holland (1975) over the course of the 1960s and 1970s and finally popularized by one of his students, David Goldberg, who was able to solve a difficult problem involving the control of gas-pipeline transmission for his dissertation (Goldberg, 1989) [HH98].

A simple GA definition:

The genetic algorithm (GA) is a model of machine learning. It mimics the evolutionary process in nature by creating a population of individuals represented by chromosomes. These individuals go through a process of evolution. Different individuals compete for resources in the environment. The "fittest" individual survives and propagates its genes. The "crossover" operation is the process of exchanging chunks of genetic information between a pair of chromosomes. As in natural evolution process, GAs also define a "mutation" process, where a gene undergoes changes in itself. A general scheme for a simple GA is shown in Table 1.1.

```
t :=0;
initialize_population(t);
evaluate_population(t);
t := t + 1;
p1 := select_parents(t);
recombine(p1);
mutate(p1);
evaluate_population(p1);
new_population := survivors (p,p1);
end;
```

Table1.1 A Simple Genetic Algorithm

In GAs, the parameters of the search space are encoded in the form of strings (called *chromosomes*). A group of individuals that interact (breed) together is called a *population*. Initially, a random population is created, which represents different points in the search space. *Fitness* is a value associated with a chromosome that assigns a relative merit of that chromosome. An objective and fitness function as associated with each string that represents the degree of goodness of the string. Based on the principle of survival of the fittest, a few of the strings are selected and each is assigned a number of copies that go into the mating pool. Biologically inspired operators like *crossover* and *mutation* are employed to form a new chromosome from parent chromosomes by combining part of the information from each. The process of selection, crossover and mutation continues for a fixed number of generations or until a termination condition is satisfied.

1.3.2 Where to Use a GA?

Genetic algorithms are randomized search and optimization techniques guided by the principles of evolution and natural genetics, having a large amount of implicit parallelism. GAs perform search in complex, large and multi-modal landscapes, and provide near-optimal solution for objective or fitness function of an optimization problem. Some of the advantages of a genetic algorithm include that it

- Optimizes with continuous or discrete parameters,
- Does not require derivative information,
- Simultaneously searches from a wide sampling of the cost surface,
- Deals with a large number of parameters,
- Is well suited for parallel computers,
- Optimizes parameters with extremely complex cost surfaces; they can jump out of a local minimum,
- Provides a list of optimal parameters, not just a single solution,
- May encode the parameters so that the optimization is done with the encoded parameters, and
- Works with numerically generated data, experimental data, or analytical function.

Genetic algorithms have rapidly gained acceptance in the scientific community, and their basic principles are now quite well known. Genetic algorithms (GAs) are efficient search methods based on principles of natural selection and genetics. There are many tasks involved in the process of analyzing / identifying a pattern which need appropriate parameter selection and efficient search in complex spaces in order to obtain optimum solutions. The application of GAs for solving certain problems of pattern recognition (which need optimization of computation requirements, and robust, fast and close approximate solution) appears to be appropriate and natural.

Genetic algorithms have been used in many pattern recognition and image processing applications including image segmentation, feature selection, and shape analysis. Further, they are being applied successfully to find acceptable solutions to problems in business, engineering, and science.

1.3.3 Co-Evolution: Competitive VS. Cooperative

Coevolutionary algorithms have received increased attention in the past few years within the domain of evolutionary computation. The co-evolution algorithm consists of an iterative loop of transmission, variation, and selection of individuals with certain traits. Because co-evolution of good design obviously occurs naturally, the co-evolution algorithm is equated to the formalization of design synthesis. Modification to the co-evolution for use on computational substrate results in the co-evolution framework that can be and is used for automatic design synthesis.

Simplistically speaking, one can say that coevolving species can either compete (e.g., to obtain exclusivity on a limited resource) or cooperate (e.g., to gain access to some hard-to-attain resource).

The problem of the cooperative design chosen for either competitive or cooperative is subject to a dynamic environment. They must be able to change with the environment. Those that have a better ability to co-evolve can evolve more quickly in concert with the dynamic environment and thus have a better chance for continued survival. In addition, dynamic environments in nature have resulted in the evolution of adaptability, or the ability of an individual to adapt itself to an environment within its life span.

Competitive Coevolution:

In a competitive coevolutionary algorithm, the fitness of an individual is based on direct competition with individuals of other species, which in turn evolve separately in their own populations. Increased fitness of one of the species implies a diminution in the fitness of the other species. This evolutionary pressure tends to produce new strategies in the populations involved so as to maintain their chances of survival. This "arms race" ideally increased the capabilities of each species until they reach an optimum level.

Predator-prey relations are the most well-known example of competitive coevolution. There is a strong evolutionary pressure for prey to defend themselves better (e.g., by running faster, growing bigger shields, better camouflage, etc.) in response to which future generations of predators develop better attacking strategies (e.g., stronger claws, better eye-sight, etc.). In such *arms races*, success on one side is felt by the other side as failure to which one must "respond" in order to maintain one's chances of survival. This, in turn, calls for a reaction of the other side. This process of coevolution can result in a stepwise increase in complexity of both predator and prey.

Cooperative Coevolution:

Cooperative coevolution refers to the simultaneous evolution of two or more species with coupled fitness. Such coupled evolution favors the discovery of complex solutions whenever complex solutions are required and widen the range of applications of evolutionary computation. Cooperative coevolutionary algorithms involve a number of independently evolving species which together form complex structure, well suited to solve a problem. The fitness of an individual depends on its ability to collaborate with individuals from other species. In this way, the evolutionary pressure stemming from the difficulty of the problem favors the development of cooperative strategies and individuals [PS01].

1.3.4 Review of Previous Work in Using GA for Pattern Clustering and Classification

A GA-clustering methodology was proposed in [MB00] where genetic algorithms (GAs) were used for searching for the appropriate cluster centres such that a similarity metric of the resulting clusters was optimized. The chromosomes, which were represented as strings of real numbers, encoded the centres of a fixed number of clusters. It demonstrated that the GA-clustering algorithm provided a performance that was significantly superior to that of the K-means algorithm in providing optimal clusters, by considering several artificial and real-life data sets with the number of dimensions ranging from two to ten and the number of clusters ranging from two to nine.

Another genetic clustering algorithm was proposed [TY01] in search automatically for a proper number as the number of cluster and classify the objects into these clusters at the same time. In searching a good clustering, the value of a parameter varied within a given range and several possible clustering were obtained. Then, a heuristic strategy was applied to choose a good clustering. Before using the genetic clustering algorithm, the single-linkage algorithm was applied to reduce the size of the data set if the size was large.

In [Cus98], GAs were used off-line for providing the training set for a learning-by-example classifier. It presented a genetic algorithm for clustering objects in images based on their visual features, especially in automated visual inspection where clustering was applied for dividing similar objects into a given number of similarity or tolerance classes. The algorithm was evaluated on some test sets, and GAs was found to be very efficient and very robust with respect to the noisy and imprecise feature measurements.

The study of using GAs to design a multiple-classifier system was proposed in [KJ00], which was more accurate than the best individual classifier in the system. The main message of this study was that GAs and other evolutionary algorithms offered an intuitive *automatic* way to design multiple-classifier systems instead of picking the classifier types and selecting the combination ad hoc by the designer. It considered the selection of features for a preliminary fixed (small) number of classifiers, and the selection of classifier types from a (small) set of basic models. Its GA-based design scheme could be extended straightforwardly to a higher number of classifiers ($L > 3$) and classifier models.

1.3.5 Review of Previous Work in Using GA for Feature Selection

Since fast processing a large volume of data is critical in application for a real-time processing or for a quick response to users, using a small number of features is a very important requirement. The idea of using multiple sets of features needs the feature selection as a prerequisite for the subsequent processing of classification or clustering proposed in [OLM02][RPG00].

A direct approach to using GAs for feature selection was introduced by Siedlecki and Sklansky [SS89]. In their work, a GA was used to find an optimal binary vector, where each bit was associated with a feature. If the *i*th bit of this vector equaled 1 then the *i*th feature was allowed to participate in classification; if the bit was a 0, then the corresponding feature did not participate. Each resulting subset of features was evaluated according to its classification accuracy on a set of testing data using a nearest neighbor classifier.

The paper [OLM02] proposed a hybrid genetic algorithm for the feature selection. It revealed better convergence properties and gave two desirable effects of significant improvement of the final performance and the acquisition of subset-size feature control. Local search operations were devised and embedded in hybrid GAs to fine-tune the search. The operations were parameterized in terms of the fine-tuning power, and their effectiveness and timing requirement were analyzed and compared. A method for rigorous timing analysis was developed, and a bookkeeping scheme was proposed for accelerating the conventional and the proposed algorithms.

An approach [RPG00] was presented to feature extraction in which feature selection, feature extraction, and classifier training were performed simultaneously using a genetic algorithm. The genetic algorithm optimizes a vector of feature weights, which were used to scale the individual features in the original pattern vectors in either a linear or a nonlinear fashion. A Masking vector was also employed to perform simultaneous selection of a subset of the features. It employed the technique in combination with the k-nearest neighbor classification and genetic algorithms was employed in feature dimensionality reduction.

1.4 Scope and Organization of the Thesis

In this thesis, a new adaptive technique of finger upright reorientation is introduced by using *the Principle of Coordinate System Rotation* in order to minimize the variance of hand images from trials to trials. A set of hand geometric features have been identified that can be used to represent a person's hand. These features include the lengths and widths of the fingers at various locations. Hand shape analysis, included Central Moments, Fourier Descriptors and Zernike Moments, is characterized based on 1-D contour transformation. In the processing of finger upright reorientation, fingers become isolating multi-contour patterns. The purpose of contour 1-D transformation served in the finger shape analysis is to improve image discrimination for the finger orientation-dependent in the projections of X-Y plane.

A new approach for Cooperative Co-evolution Clustering Algorithms (CCCA) which integrates dynamic clustering and feature selection for hand images is proposed. CCCA is defined as the problem of classifying a collection of hand objects into a set of natural clusters without any a priori knowledge with feature dimension reduction. The dimensionality reduction results in a performance improvement, especially when the size of the training set is small compared to the feature vector size. In addition to the main contribution of the study, the *MSE Extended Fitness Function* is presented which is particularly suited to an integrated dynamic clustering space.

The next Chapter of the thesis is organized as follows. In Chapter 2, the emphasis is placed on a new design of finger upright reorientation as the primary contribution of this thesis, and the experimental results of this design are presented. The methods of hand geometry feature extraction are described in Chapter 3. It includes hand shape analysis using Central moments, Fourier Descriptors and Zernike Moments extensively. The objective of Contour 1-D transformation design is to alleviate the problems of isolated upright fingers for their orientated-dependence in the projections of X-Y plane. In Chapter 4, a new approach of Cooperative Coevolutionary Clustering Algorithm in dynamic clustering and feature selection is presented. The proposed design and implementation are applied to the problem of searching for a proper number of clusters and grouping the hand objects into these clusters to perform simultaneous selection of a subset of the features. Finally, results, discussion and future works are given in Chapter 5.

Chapter 2

Finger Reorientation

2.1 Finger Reorientation Purpose

Hand features play an important role in image recognition. It has three goals in the design of pattern classifiers:

(1) to reduce the cost of extracting features,

(2) to improve the classification accuracy, and

(3) to improve the reliability of the estimate of performance [KS00].

To compensate for the variable position of the hand from trial to trial (because there was no device to hold hand in a fixed position for the hand enrollment), an adaptive method of finger reorientation is applied to align the fingers vertically before feature extraction. The purpose is to align each finger prior to any feature extraction in order to minimize the variation of hand placements in data entry and improve the quality of feature representation.

2.2 Reorientation Algorithms

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

The line $\overline{AB}$ connecting the minima of the two phalanges *A* and *B* has a midpoint: call it finger base-line midpoint, or *Om*. The line $\overline{CO_m}$ connecting *Om* point with the finger centre point *C* makes an angle θ with the vertical Y-axis or Y'-axis. 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 θi.

The upright procedures to find the finger rotation line $\overline{CO_m}$ are as follows, shown in Figure 2.1:

1) get the minima of two phalange points *A* and *B*;
2) connect points A and B, and get the middle point $O_m(X_m,Y_m)$ of the line $\overline{AB}$;
3) find the finger midpoint center *C* in a finger triangle ΔTAB;
4) connect two points *C* and O_m, and define the line $\overline{CO_m}$ as the finger rotation line.

The point $O_m(X_m,Y_m)$ is chosen as a new original of the coordinate system for X'-Y' plane, served as the rotation point, as shown in Figure 2.2. In this way, its coordinate system transformation from X-Y plane to X'-Y' plane is:

$$X' = X - X_m$$
$$Y' = Y - Y_m$$

In its new original $O_m(X_m,Y_m)$ of coordinate system X'-Y' plane, finger rotates with its rotation angle θ given by Y'-axis and Y''-axis, or by the rotation line $\overline{CO_m}$ and

Y'-axis. According to *the Principle of Coordinate System Rotation*, the rotation formulae are described as follows:

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

X: principal x-axis, X' is the axis after the first transformation and X'' is x-axis after the second (and final) transformation;
Y: principal y-axis, Y' is the y-axis after the first transformation and Y'' is y-axis after the second (and final) transformation;
$O_m(X_m, Y_m)$: a new original of coordinate system X'-Y' plane, served as the rotation point.
$O\ (X_o, Y_o)$: the original of coordinate system X-Y plane.
θ: the rotation angle. If the plane is rotated clockwise, $\theta >= 0$. Otherwise, $\theta < 0$.

Then, because of different horizontal levels of the rotation point $O_m(X_m,Y_m)$ for each finger, the relatively upright fingers are moved horizontally and vertically to set all of them above Y''= 0 and isolate each finger along the X'' axis.

A pseudo code for upright algorithm is provided in the Appendix.

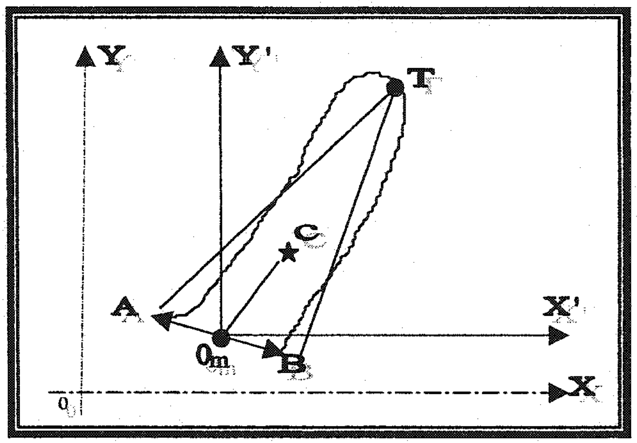

Figure 2.1 Reference rotation lines $\overline{CO_m}$: the finger triangle ΔTAB is given by the finger tip *T* together with the minima of the two phalanges on either side of that finger *A* and *B;* the point *C* is the finger midpoint centre; the finger base-line midpoint *Om* is the midpoint of the line $\overline{AB}$; the line $\overline{CO_m}$ makes a rotation angle θ with the vertical Y'-axis.

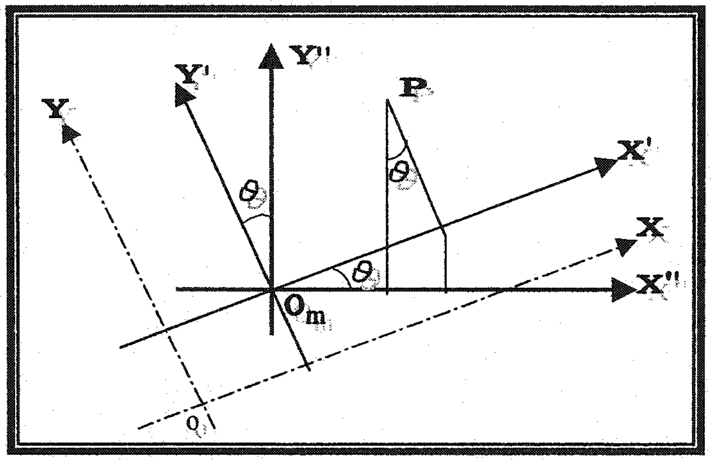

Figure 2.2 Coordinate system rotated with an angle θ from X-Y plane to X"-Y" plane. θ is given by Y'-axis and Y"-axis, or by Y'-axis and the rotation line $\overline{CO_m}$. The rotation formulae are described as follows:

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

2.3 Finger Reorientation Experiments

The performance of the finger upright, shown in Figure 2.3, with their rotation angles in Table 2.1, is analyzed by the statistic measurement of computing numerical descriptive measure Coefficient of Variance (CV), to compare to the features before finger upright. CV is a relative measure of variability to distinguish the relative variability of data sets. The smaller values of CV represent that the features have relatively lower variability to the mean [KW00].

Features with good stability and reliability should exhibit a small CV among the samples for the same hand. The upright experiment results, shown in Table 2.2, indicate that the alteration of finger uprightness improves effectively the representation of feature extraction as to achieve better performance for object recognition. They also show that reorientation consistently leads to more stable feature values, regardless of the features measured. This is a significant finding. Hence, it is highly recommended that before any feature extraction the various fingers have to be made to stand upright.

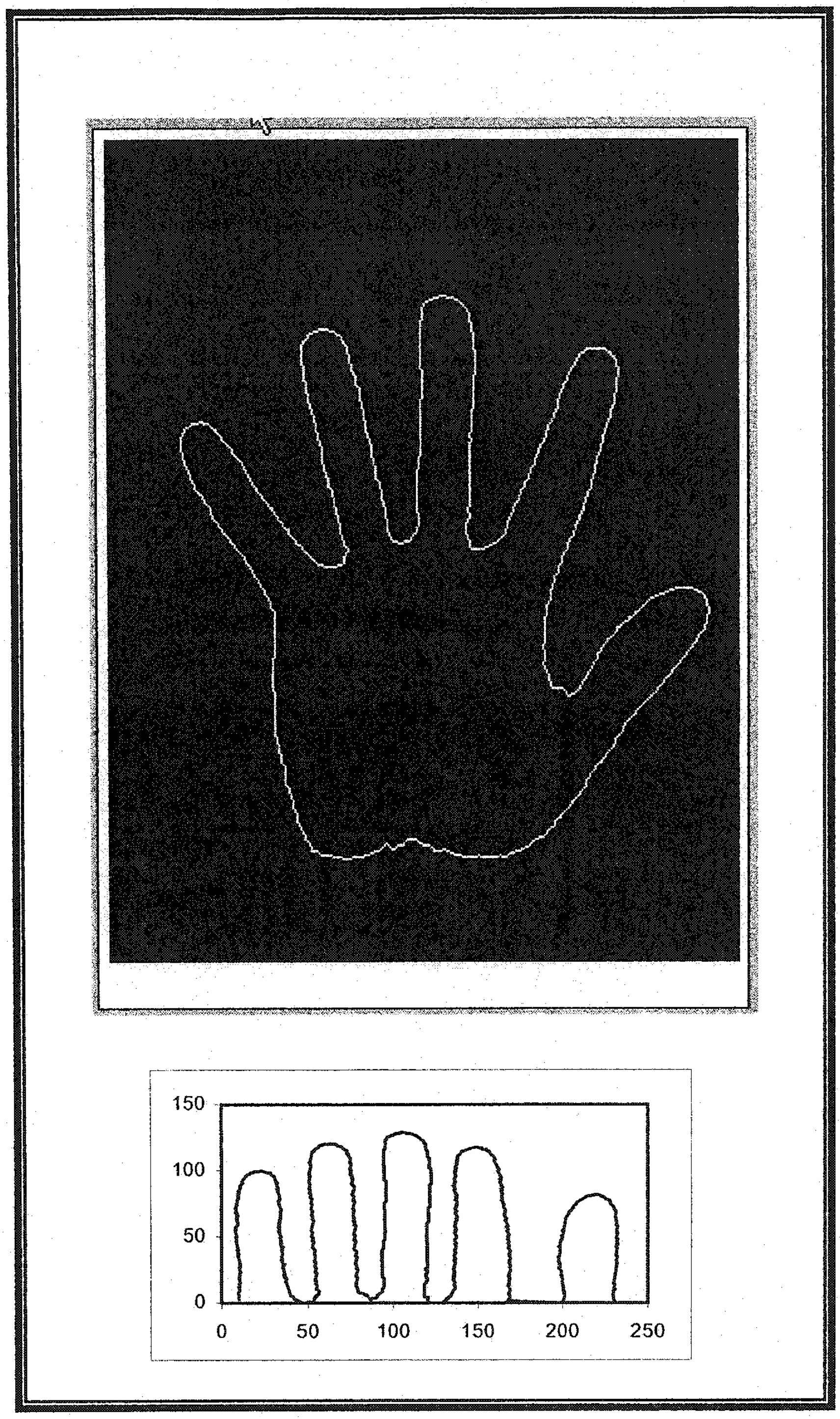

Figure 2.3 Finger reorientation

Table 2.1 Finger Rotation Angles

	Fingers				
	Little finger	Ring finger	Middle finger	Index finger	Thumb
Rotation Angles	31.47°	10.35°	-0.674°	-22.07°	-37.39°

Table 2.2 Feature Measurement After Finger Reorientation

	Finger Before Upright						Finger After Upright					
CENTRAL	S1	S2	S3	S4	S5	CV(%)	S1	S2	S3	S4	S5	CV(%)
$M1$	0.2884	0.2937	0.2917	0.3033	0.2997	1.830	0.2873	0.2899	0.2901	0.3018	0.2901	1.750
$M2$	0.1384	0.1394	0.1416	0.1429	0.1460	1.910	0.1374	0.1389	0.1403	0.1429	0.1375	1.470
$M3$	0.1111	0.1123	0.1136	0.1151	0.1171	1.830	0.1103	0.1115	0.1125	0.1147	0.1109	1.390
$M4$	0.1126	0.1159	0.1136	0.1198	0.1165	2.180	0.1120	0.1133	0.1128	0.1177	0.1148	1.770
FOURIER	S1	S2	S3	S4	S5	CV(%)	S1	S2	S3	S4	S5	CV(%)
$A1$	0.2079	0.2052	0.2150	0.2079	0.2246	3.310	0.2039	0.2050	0.2055	0.2074	0.2085	0.810
$A2$	0.1006	0.0982	0.1018	0.1009	0.1031	1.610	0.1000	0.0977	0.1003	0.1011	0.0997	1.120
$A3$	0.0664	0.0646	0.0671	0.0667	0.0682	1.780	0.0660	0.0642	0.0661	0.0668	0.0653	1.330
$A4$	0.0497	0.0484	0.0503	0.0500	0.0512	1.850	0.0494	0.0481	0.0494	0.0500	0.0490	1.290
$A5$	0.0396	0.0385	0.0401	0.0398	0.0407	1.830	0.0393	0.0383	0.0393	0.0398	0.0390	1.280
ZERNIKE	S1	S2	S3	S4	S5	CV(%)	S1	S2	S3	S4	S5	CV(%)
Z_{20}	0.3300	0.3429	0.3359	0.3415	0.3277	1.800	0.3295	0.3391	0.3352	0.3395	0.3260	1.590
Z_{22}	0.0829	0.0868	0.0849	0.0878	0.0819	2.620	0.0828	0.0850	0.0847	0.0869	0.0819	2.100
Z_{31}	0.4180	0.4345	0.4285	0.4432	0.4128	2.570	0.4175	0.4266	0.4277	0.4391	0.4123	2.170
Z_{33}	0.0909	0.0940	0.0935	0.0975	0.0895	2.970	0.0908	0.0921	0.0933	0.0965	0.0896	2.570
Z_{42}	0.5300	0.5450	0.5464	0.5696	0.5211	3.050	0.5296	0.5350	0.5458	0.5640	0.5213	2.740
Z_{44}	0.0981	0.1003	0.1014	0.1060	0.0963	3.300	0.0981	0.0985	0.1013	0.1050	0.0964	3.020
GEOMETRIC	S1	S2	S3	S4	S5	CV(%)	S1	S2	S3	S4	S5	CV(%)
LENGTH	106.0	101.3	106.3	106.7	106.1	1.920	106	103.3	106.3	107.4	106.5	1.320
MIDLINE	113.8	112.7	112.8	113.1	110.6	0.926	114.7	112.9	115.2	113.8	114.3	0.705
TIP ANGLE	18.02	17.86	19.08	17.35	19.45	4.290	18.02	17.75	19.08	17.43	19.11	3.780
HEIGHT	110.4	111.7	110.4	111.5	108.6	0.991	111.8	111.6	112.7	111.5	112.4	0.422

Si:	*The ith Sample.*
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 σ divided by their mean μ for samples.

Chapter 3

Feature Extraction

3.1 Introduction

The first stage in any pattern recognition task is usually referred to as *feature extraction.* Feature extraction is nothing more than a process of measurement, but often this process can be so complicated that it constitutes the main task of the pattern recognition. The result of the feature extraction stage is a set of numbers x that are fed to the classification or decision stage of the recognition [Jam88].

Feature extraction plays an important role in image identification and verification. In the feature extraction phase, only the salient features necessary for the recognition process are retained such that the classification can be implemented in a vastly reduced feature space. Feature extraction changes the data into a form that is simpler, and easier for the system to learn the target concept, resulting in a simpler hypothesis.

Moments and functions of moments have been utilized as shape analysis in pattern recognition. Such features capture global information about the image and do not require closed boundaries as boundary-based methods. Shape analysis has played an important role in pattern recognition since the origin of the field in the 1960s. Hu (1962) proposed a method of deriving moment invariants from algebraic invariants and listed examples of different types of moment invariants. Abu-Mostafa and Psatis (1984) investigated the recognition properties of moment invariants. They studied the influence factors and robustness of moment invariants. Moment functions such as geometric moments, rotational moments and complex moments are all useful tools in the field of pattern recognition and can be used to describe the features of objects such as the shape, area, border, location and orientation. Moment invariants are invariant under shifting, scaling and rotation. They are widely used in pattern recognition because of their discrimination power and robustness.

Invariant descriptions can be measured directly from objects without detailed prior knowledge of the projective, or other transformations. In many cases, one is only interested in those properties that are invariant under a particular class of transformations. The fundamental difficulty in recognizing an object from its image is that the appearance of shape depends on the viewpoint.

For shape analysis, three fundamental issues related to their usefulness must be considered. They include (1) sensitivity to image noise, (2) aspects of information redundancy and (3) capacity for image representation. One important advantage of using moments for shape analysis is that it is relatively insensitive to image noise.

Selection of feature extraction methods is probably the single most important factor in achieving high recognition performance in a pattern recognition system. Positive identification, based on hand feature extraction, is a challenging pattern recognition problem in its own right.

3.2 Geometric Features

Biometrics-based authentication is a verification approach using the biological features inherent in each individual. Biometric features have been widely used in many personal authentication applications. They are based on the identical, portable and arduous duplicate characteristics.

Hand Geometry, as the name suggests, refers to the geometric structure of the hand. There are many features exhibited in a hand. The structure includes height of the fingers, angles of the fingers at various locations and width of the palm.

The hand structure we measured can be divided into six categories:

1) Finger width(s):

   The distance between the minima of the two phalanges on either side of a finger, shown in Figure 3.1. They are X_0X_1, X_1X_2, X_2X_3, X_3X_6 and X_4X_5.

2) Finger height(s):

   The length of the line starting at the fingertip and intersecting (at right angles) with the finger base-line, shown in Figure 3.1.

3) Finger circumference(s):

   The length of the finger contour.

4) Finger angle(s):

   In each finger, there are three typical points, the tip and two phalange joints, which create a finger triangle. Figure 3.2 shows an example of the finger triangle $\Delta T_2X_2X_3$ in the middle finger. Two angles are measured inside each finger triangle: a tip angle $\angle X_2T_2X_3$ and a phalange angle $\angle T_2X_2X_3$.

5) Finger base length(s):

The length of the finger base-lines, shown in Figure 3.3. They are X_1X_4, X_2X_4, X_3X_4, X_6X_4 and X_1X_3.

6) Palm aspect ratio:

The ratio of the 'palm width' to the 'palm height'. Palm height is (double) the distance between the minima X_2 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 M_p). Palm width is (double) the shortest distance between M_p and the right edge of the palm image.

As shown in Figure 3.4, the procedure to get the palm aspect ratio is as follows:

(i) connect two phalange points X_0 and X5;

(ii) get the midpoint M_p in the line $\overline{X_0 X_5}$;

(iii) from point M_p , find the closest point P in the outer palm ridge;

(vi) divide the length $\overline{X_2M_p}$ by the length $\overline{PM_p}$ to get the ratio of palm height to palm width.

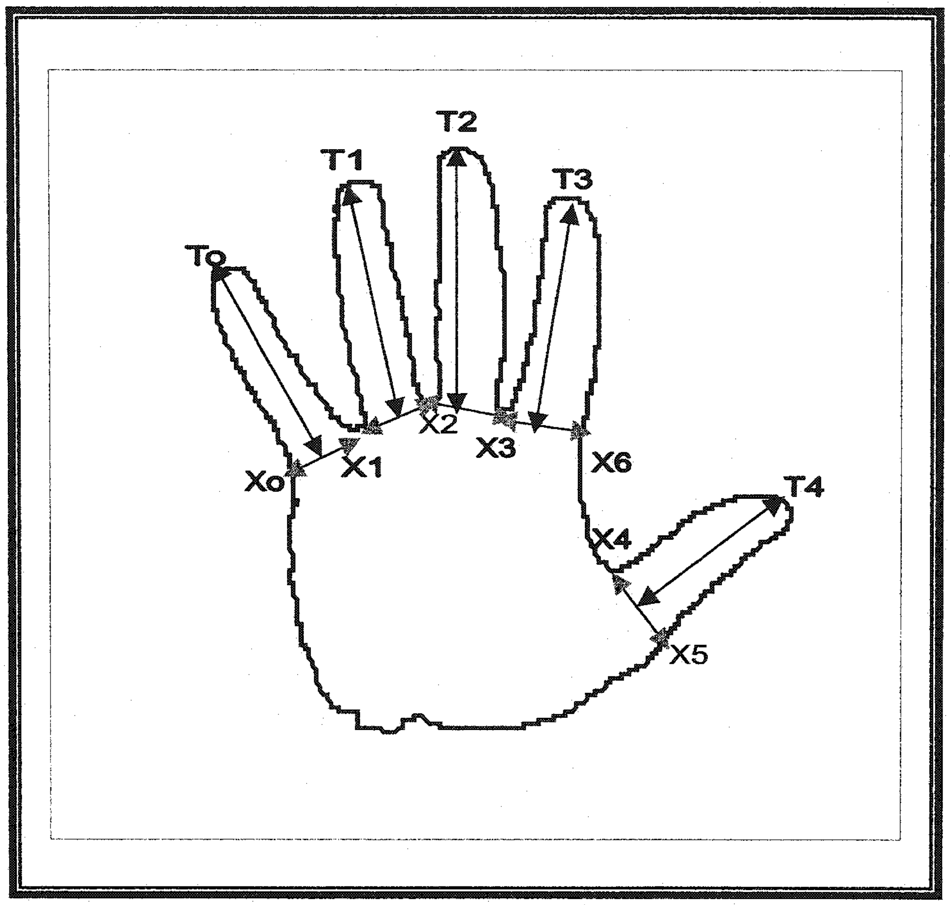

Figure 3.1 Features of finger widths and heights

$T_0 - T_4$:	*Finger tips*
$X_0 - X_6$:	*Finger phalange joints*

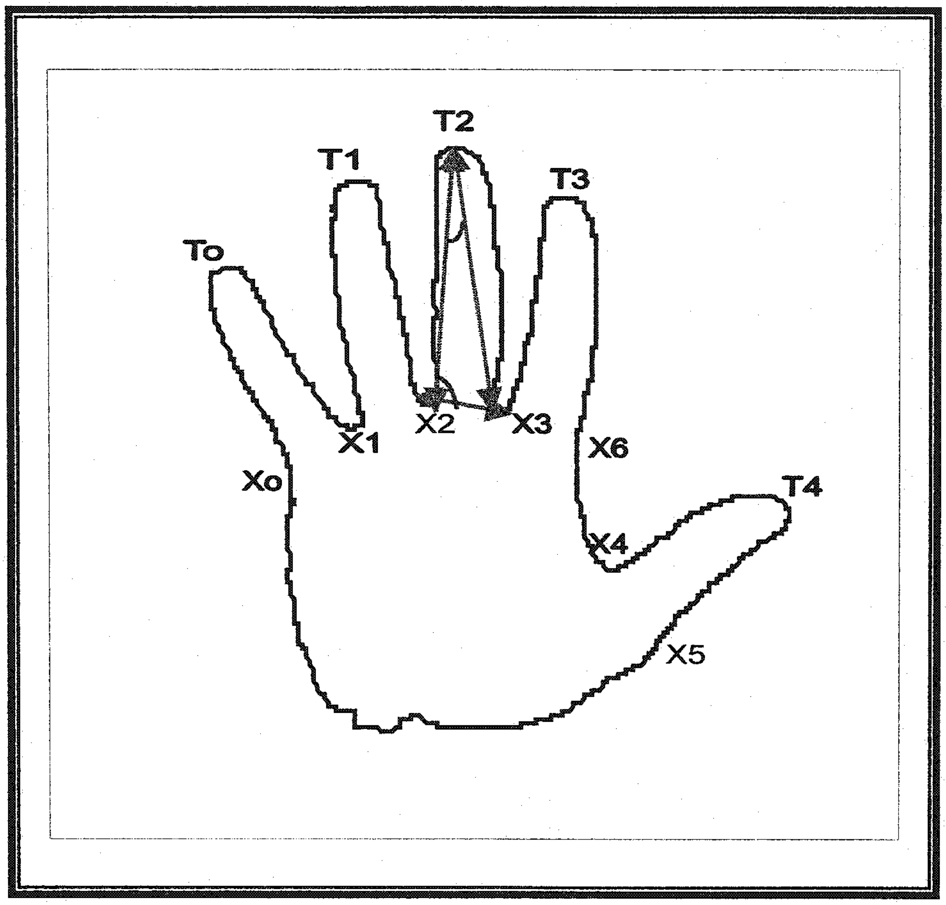

Figure 3.2 Features of a tip angle $\angle X_2T_2X_3$ and a phalange angle $\angle T_2X_2X_3$ in a finger triangle $\Delta T_2X_2X_3$

T_0 -- T_4:	*Finger tips*
X_0 -- X_6:	*Finger phalange joints*

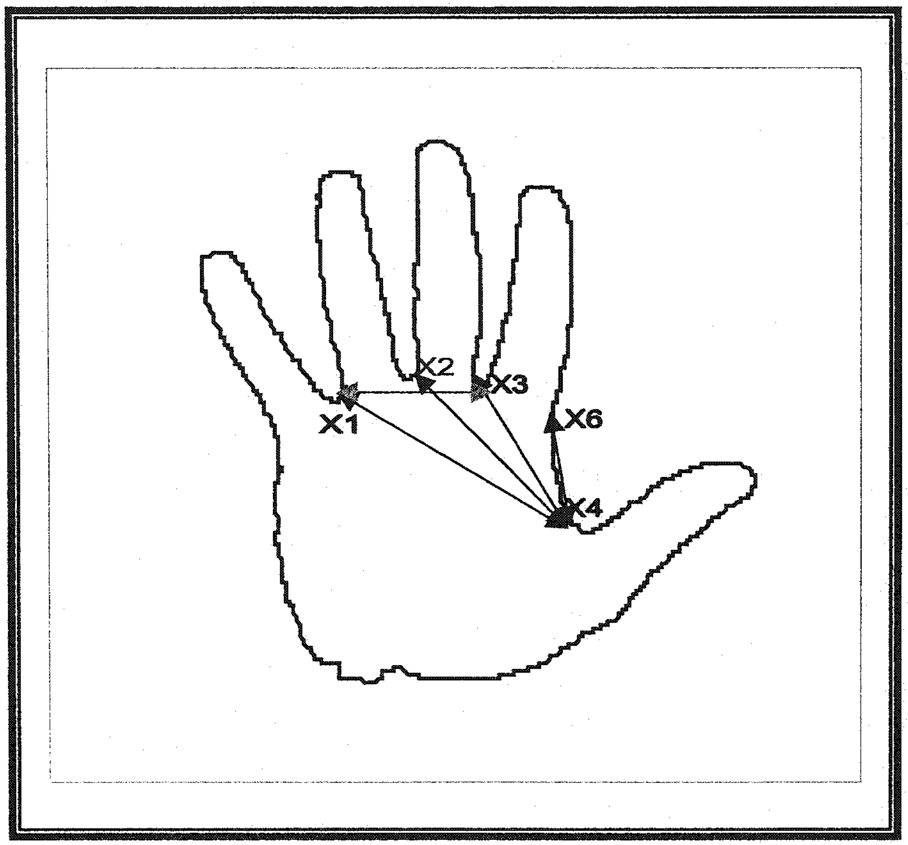

Figure 3.3 Features of finger base lengths: X_1X_4, X_2X_4, X_3X_4, X_6X_4 and X_1X_3

X_1 --X_6 :	*Finger phalange joints*

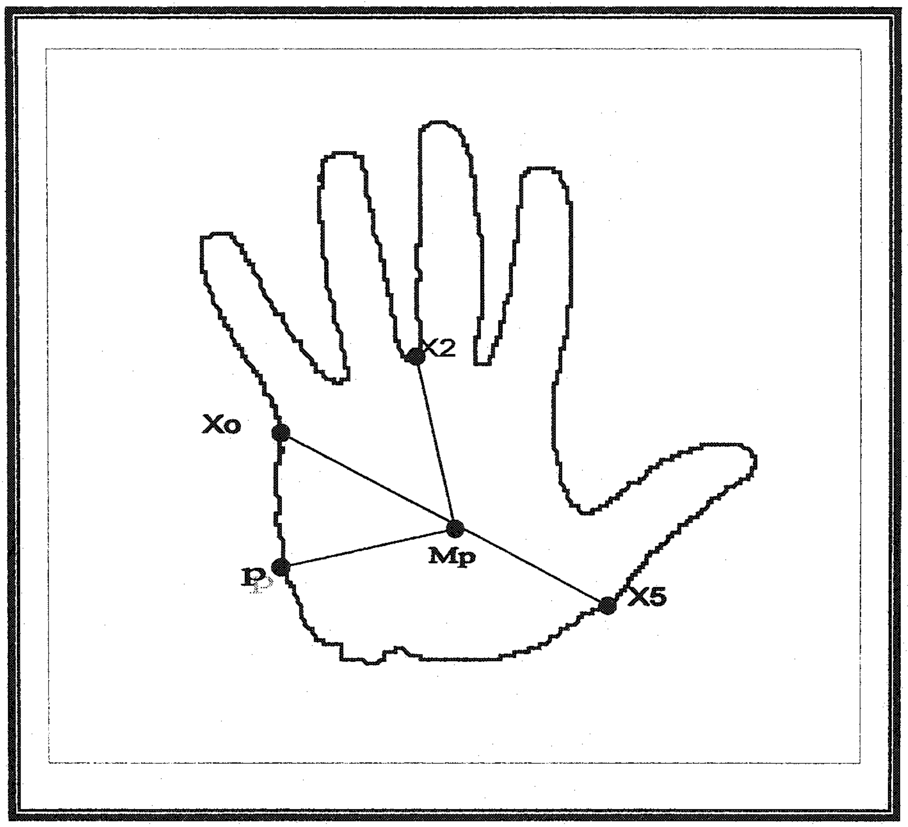

Figure 3.4. Palm aspect ratio: X_0, X_2 and X_5 are the minima of finger phalanges; M_p is the midpoint of the line $\overline{X_0X_5}$; point P is the shortest distance between M_p and the right edge of the palm; the palm aspect ratio equals to the line $\overline{X_2M_p}$ divided by the line $\overline{PM_p}$

X_0 - X_5 :	*Finger phalange joints*
M_p :	*Midpoint of XoX_5*
P :	*The closest point from point M_p to the palm ridge*

3.3 Shape Analysis

For image shape analysis, many methods such as chain code, polygonal approximation, moments, Fourier descriptors, etc., can be used. For invariant shape recognition, we can use such approaches as geometrical moment and Zernike moment, boundary-based analysis via Fourier descriptors.

Each finger has its own characteristics in hand shape analysis. After finger upright, fingers are multi-contour patterns, each of fingers composes of several isolated parts and the finger values are orientation-dependent in the projections of X-Y plane. To alleviate these problems, contour 1-D transformation is developed before shape analysis.

Before doing shape analysis, we mapped each finger 2-D contour into 1-D contour [inspired by, but still significantly different from [WBM98]], taking the finger midpoint center as its reference point.

Since selection of good features is a crucial step of the analysis, a flexible recognition system must be able to recognize an object regardless of is orientation, size and location in the field of view, i.e., rotation-, scale- and translation-invariance. For shape analysis for four fingers, excluding the thumb, we measured after fingers upright using: (1) Central moments; (2) Fourier descriptors; (3) Zernike moments.

3.3.1 Finger 1-D Contour Transformation

Once the fingers are reoriented, such that each finger is separated and vertically aligned, a 2D-to-1D mapping is carried out. The 2D contour of each finger is mapped onto a 1D graph (see Figure 3.5). 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, served as the reference point, 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 assigned to be the Euclidean distance between these points and the reference point.

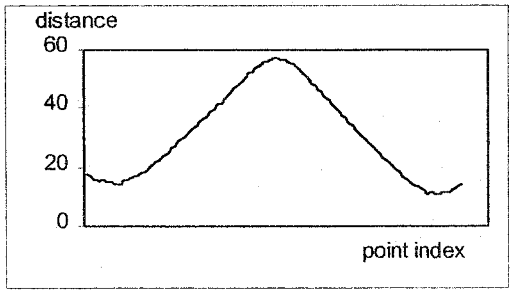

Figure 3.5 1D contour of a finger

Assuming the obtuse angle corresponding to the 1-D contour curvature is divided into equal angles formed by the total 140 points, the length of vector between the reference point and index can be described in polar form $f[\rho_j, \varphi_j]$ for the *j*th pixel.

3.3.2 Central Moments

Moments approach is quite often used to describe a shape of an object. Central moments have by far been the popular type of moments.

For a digital image, the *p*th order regular moment with one-dimensional function *F[n]* is defined as follows:

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

The normalized one-dimensional *p*th order central moment is defined as follows:

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

where $\bar{n} = R_1 / R_0$

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

Table 3.1 shows the values of Central moments.

3.3.3 Fourier Descriptors

Invariant descriptors are properties of geometric configurations, which remain unchanged under an appropriate class of transformations. Define a normalized cumulative function Φ* as expanding Fourier series to obtain descriptive coefficients that we call Fourier Descriptors (FD's).

Given a periodic 1-D digital function *F[n]* in a periodic [0, N] points, expanding Fourier series:

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

where

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

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

The *k*th harmonic amplitude of Fourier Descriptors:

$$A_k = \sqrt{a_k^2 + b_k^2} \qquad k = 1,2,\ldots\ldots$$

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

Table 3.2 shows the values of Fourier descriptors.

3.3.4 Zernike Moments

Zernike moments are a class of orthogonal moments. It outperforms the others (Belkasim et al., 1991, pp. 1117-1138) generally even though it is more complicated than the geometrical moment. For Zernike moments, Zernike introduced a set of complex polynomials which form a complete orthogonal set over the interior of the unit circle of $x^2 + y^2 = 1$. The Zernike polynomial can be used for reconstruction and recognition of shape.

To compute the Zernike moments of a given image, the center of the image is taken as the origin and the pixel coordinates are mapped to the range of the unit circle. Those pixels falling outside the unit circle are not used in the computation.

For a digital image with a polar form function $f(\rho_j, \varphi_j)$, the normalized $(n+m)$th order Zernike moment is approximated by

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

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

$$x_j^2 + y_j^2 \leq 1,$$

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

n: a positive integer.
m: a positive and negative integer subject to constraints n-|m| is even, |l|<=n.
N: the total number of pixels.
$f(\rho_j, \varphi_j)$*: the pixel value of the jth pixel.*
'' denotes the complex conjugate.*

Table 3.3 shows the values of Zernike moments.

Table 3.1 Central Moments

Central Moments										
	S1	S2	S3	S4	S5	CV(%)	Mean	Range	Maximum	Minimum
M1	0.3079	0.3091	0.3089	0.3105	0.3106	0.335	0.3094	0.00273	0.3106	0.3079
M2	0.1485	0.1486	0.1485	0.1492	0.1492	0.224	0.1488	0.00073	0.1492	0.1485
M3	0.1165	0.1170	0.1169	0.1176	0.1177	0.383	0.1171	0.00119	0.1177	0.1165
M4	0.1118	0.1134	0.1134	0.1143	0.1146	0.872	0.1135	0.00282	0.1146	0.1118

Si: The ith Sample.
Mk: The kth order central moments.
CV(%): Equal to the standard deviation of the measurements divided by their mean for samples.
Mean: The average value of the samples.
Range: Range between the maximum and minimum values of the samples.
Maximum: The maximum value among the samples.
Minimum: The minimum value among the samples.

Table 3.2 Fourier Descriptors

Fourier Descriptor										
	S1	S2	S3	S4	S5	CV(%)	Mean	Range	Maximum	Minimum
A1	0.2100	0.2094	0.2091	0.2097	0.2095	0.145	0.2095	0.00090	0.2100	0.2091
A2	0.1029	0.1028	0.1027	0.1030	0.1029	0.113	0.1029	0.00033	0.1030	0.1027
A3	0.0681	0.0679	0.0679	0.0681	0.0680	0.124	0.0680	0.00023	0.0681	0.0679
A4	0.0509	0.0508	0.0508	0.0510	0.0509	0.140	0.0509	0.00019	0.0510	0.0508
A5	0.0405	0.0406	0.0404	0.0406	0.0406	0.144	0.0405	0.00016	0.0406	0.0404

Si: The ith Sample.
Ak: The kth harmonic amplitude of Fourier descriptors.
CV(%): Equal to the standard deviation of the measurements divided by their mean for samples.
Mean: The average value of the samples.
Range: Range between the maximum and minimum values of the samples.
Maximum: The maximum value among the samples.
Minimum: The minimum value among the samples.

Table 3.3 Zernike Moments

Zernike Moments										
	S1	S2	S3	S4	S5	CV(%)	Mean	Range	Maximum	Minimum
Z20	0.35359	0.35070	0.35000	0.34990	0.34930	0.435	0.3507	0.0043	0.35359	0.34925
Z22	0.09240	0.09130	0.09100	0.09100	0.09074	0.642	0.0913	0.0017	0.09240	0.09074
Z31	0.23390	0.23100	0.23010	0.23000	0.22950	0.674	0.2309	0.0044	0.23390	0.22950
Z33	0.05196	0.05120	0.05097	0.05100	0.05083	0.781	0.0512	0.0011	0.05196	0.05083
Z42	0.10160	0.10000	0.09950	0.09960	0.09920	0.835	0.1000	0.0023	0.10157	0.09922
Z44	0.01901	0.01870	0.01859	0.01861	0.01854	0.900	0.0187	0.0005	0.01901	0.01854

Si: *The ith Sample.*
Znm: *The (n+m)th order Zernike moments*
CV(%): *Equal to the standard deviation of the measurements divided by their mean for samples.*
Mean: *The average value of the samples.*
Range: *Range between the maximum and minimum values of the samples.*
Maximum: The maximum value among the samples.
Minimum: The minimum value among the samples.

Chapter 4

Cooperative Coevolution: Dynamic Clustering and Feature Selection

4.1 Introduction

For classical pattern recognition techniques, the patterns are generally represented as a vector of feature values. The selection of features can have a considerable impact on the effectiveness of the resulting classification algorithm. Feature selection takes in as input the original N features and generates as output M (where $M < N$) new features. The advantage of performing feature extraction is that it is a preprocessing process and works on data independently of the learning algorithm so that one can still use any learning algorithm one prefers, but has its performance boosted.

Feature selection can be used to project data onto a lower dimensional space for subsequent visualization, clustering, and other exploratory data analysis. By reducing the number of features considered by a classifier, the techniques can improve classification speed and efficiency. Less intuitively, these techniques can improve classification accuracy by reducing estimation errors associated with finite sample size effects.

Clustering problems generally appear in classification of data for some purpose like storage and retrieval or data analysis. Any clustering algorithm will attempt to determine some inherent or natural grouping in the data, using distance or similarity measures between individual data. For this it is necessary to first define a measure of similarity which will establish a rule for assigning patterns to the domain of a particular cluster center. One such measure of similarity may be the Euclidean distance D between two patterns x and z defined by $D = | x - z |$. The smaller the distance between x and z, the greater is the similarity between the two and vice versa.

GAs encode the concept of evolution in the algorithmic search: from a population of individuals representing possible problem solutions, evolution is carried out by means of selection and reproduction.

Although most design problems have variable complexity search spaces, nearly all GA implementations have fixed length representations or chromosomes. What this means is that the structure of the optimal design is somehow known a *priori*; yet it is almost always the case that design functionality is dependent on both complexity and parameter values. Thus, without knowing the optimal parameter values, the optimal number of parameters is also unknown and vice versa. This implies that complexity and the ensuing parameter values must be evolved simultaneously for effective search.

4.2 Formalizing Design

The designs of cooperative co-evolutionary clustering application in this thesis involve the optimization of three quantities, which together form a complete solution:

(1) the set of features (dimensions) used for clustering;

(2) the actual cluster centres; and

(3) 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. Indeed, it is the hypothesis that whenever a (complete) potential solution (i) is comprised of a number of complementary components; (ii) has a medium-high degree of dimensionality; and (iii) features a relatively low level of coupling between the various components; then attempting a cooperative co-evolutionary approach is justified.

In similarity-based clustering techniques, a number of cluster centres are proposed. An input pattern (point) is assigned to the cluster whose centre is 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 (again) 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 genetic algorithm used entails that have two populations (one of cluster centres and another of dimension selections: more on this below), each going 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 only difference between a GA applied to one (evolving population) and a GA applied to two cooperatively co-evolving populations is that fitness evaluation of an individual in one population is done after that individual is joined to another individual in the other population. Hence, the fitness of individuals in one population is actually coupled with (and is evaluated with the help of) individuals in the other population.

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 the steps in detail, and a discussion of the results is presented.

4.3 Chromosomal Representation

In multiple-populations, GAs are more sophisticated, as they consist of several sub-populations which exchange individuals occasionally. Chromosomes are evaluated in their fitness through an optimization function in order to survive to the next generation. In any co-evolutionary genetic algorithm, two (or more) populations co-evolve. This algorithm can be regarded as a parallel and randomized algorithm.

In this co-evolutionary application design, two populations (*Cpop and Dpop*) of the jointed chromosomes are maintained and evolved by two operators of crossover and mutation. There are two populations, one is a population of cluster centres (*Cpop*), each represented by a variable-length vector of real numbers; and the other is 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' bits, which dimensions will be used and which, via its '0' bits, will not be used. Splicing an individual (or chromosome) from *Cpop* with an individual (or chromosome) from *Dpop* will give us an overall chromosome that has the following form:

$$\{(A_1, B_1, \ldots, Z_1), (A_2, B_2, \ldots, Z_2), \ldots (A_n, B_n, \ldots, Z_n), 10110\ldots0\}$$

Taken as a single representational unit, this chromosome determines:

(1) The *number of clusters*, via the number of cluster centres in the left-hand side of the chromosome;

(2) The actual *cluster centres*, via the coordinates of cluster centres, also presented in the left-hand side of the chromosome; and

(3) The *number of dimensions* (or features) used to represent the cluster centres, via the bit vector on the right-hand side of the chromosome.

As an example, the chromosome presented above has n clusters in three dimensions: the first, third and fourth dimensions. (This is so because the bit vector has 1 in its first bit location, 1 in its third bit location and 1 in its fourth bit location.) The maximum number of feature dimensions (allowed in this example) is equal to the number of letters in the English alphabet: 26, while the minimum is 1. And, the maximum number of clusters (which is not shown) is $m>n$.

4.4 Crossover and Mutation, Generally

GAs use two operators loosely based on natural genetics to explore the search space: crossover and mutation. Crossover is the primary exploration mechanism in GAs. Mutation is usually considered a secondary search operator. Its function is to restore diversity that may be lost from the repeated application of selection and crossover.

Crossover:

Crossover exchanges information between two parent chromosomes and introduces new information for the child chromosomes. Each crossover operator takes the characteristics of the parents to form the chromosomes of the children.

The main purpose of crossover is to exchange information between randomly selected parent chromosomes with the aim of not losing any important information (minimum disruption of the structures of the chromosomes that are selected for genetic operation). Actually, it recombines genetic material of two parent chromosomes to produce offspring for the next generation.

The most important trait of a crossover operator is how offspring differ in, how many children are produced and the method in which they are produced. The crossover operator has the property of inheriting the good traits from parents and discovering the new traits by its own in the next generation. It also keeps the algorithm out of local minimum.

Mutation:

Mutation is the second way a genetic algorithm explores the fitness surface. It can introduce traits not in the original population and keeps the genetic algorithm from converging too fast, or stagnating in a sub-optimal local maxima.

The role of mutation is to insert new diversity into a population. This has several purposes. Small mutations may help nudge near-optimal organisms towards perfection. Homogenous populations may suffer from over-specialization (or, in the computer, convergence on a local maximum); inserting random genes may create individuals that vary wildly from the norm, allowing the overall process to move away from sub-optimal solutions. (Thus, the role of mutation is more important when dealing with small-populations, where inbreeding is likely to lead to homogeneity much faster.) Mutation is the spontaneous, random alteration of a gene. In nature, mutations are caused by a variety of factors, including oxidizing chemicals (i.e., free radicals), radiation, or defects in the DNA duplication process.

Due to its random nature, mutation is a low-yield process. In nature, most mutations are detrimental to an organism, often even lethal. Similarly, in the computer, random changes to a chromosome are likely to reduce its fitness; even more so when the organism is highly fit in the first place. Thus, mutation rates should diminish as the population converges towards a satisfactory solution. However, the occasional positive mutation is one of the key processes in evolution.

Mutation causes the search to seek out alternate routes, ideally leading to the global optimum. In the computer, mutation is simulated by flipping randomly selected bits (in binary genetic algorithms) or replacing an entire floating-point parameter by a new random value (in continuous genetic algorithms).

In this 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) to be able to adjust the values of the coordinates of the various cluster centres. This is not a trivial task, and is achieved via a host of crossover operators, each tuned to its own task.

4.4.1 Crossover and Mutation for *Cpop*

CPop needs crossover and mutation operators suited for variable-length clusters 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:

(a) 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;

(b) Each child chromosome picks up copies of cluster centre coordinates, from each of the two parents, in proportion to the relative fitness of the parents (to each other); and finally,

(c) To vary local of clusters, the range of each feature dimension between the maximum and minimum values is applied to make dimension fine-tune to change its local of clusters. The actual values of the cluster coordinates are modified using the following (mutation) formula:

$$f_i = min(F_i) + \alpha\, [max\,(F_i) \; - \; min\,(F_i)] \qquad i=1,2,\ldots$$

F_i:	*the ith feature dimension*
$min(F_i)$:	*minimum value that feature i can take*
$max(F_i)$:	*maximum value that feature i can take*
α:	*randomly selected from the range [0,1]*

With α changed within [0,1], the function of equation above varies the *i*th feature dimension in its own feature distinguished range [$min(F_i)$, $max(F_i)$] as for the variation of actual values of the cluster coordinates (see Figure 4.1).

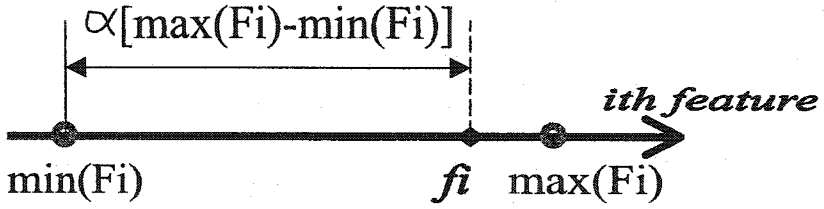

Figure 4.1 Variation of the *ith* feature dimension within [$min(F_i)$, $max(F_i)$] with a random value α ranged [0,1]

In addition to crossover, mutation is applied, with a probability μ_c to one set of cluster centre coordinates. The value of μ_c used was 0.02 (or 2%).

4.4.2 Crossover and Mutation for *Dpop*

Crossover is a reproduction operation, which forms a new chromosome from parents by combining parts of each parent chromosomes. Broadly speaking, any number of parents may be chosen to form any number of children depending on the kinds of crossover. For example, a simplest form of binary crossover is *single-point crossover*, in which an arbitrary point in the chromosome is picked. All the information from parent A is copied from the start up to the crossover point, then the information from parent B is copied from the crossover point to the end of the chromosome.

Often the individuals are composed of a binary string of a fixed length, *l*, and thus GAs explore a search space formed by 2^l points. Initially, the population consists of points chosen randomly, unless there is a heuristic to generate good solutions for the domain. In the latter case, a portion of the population is still generated randomly to ensure that there is some diversity in the solutions.

Single-point crossover:

This operator takes two random individuals from those already selected to form the next generation and exchanges random substrings between them. As an example, consider strings A_1 and A_2 of length 8:

$$A_1 = 0\ 1\ 1\ 0 \mid 1\ 1\ 1\ 1$$

$$A_2 = 1\ 1\ 1\ 0 \mid 0\ 0\ 0\ 0$$

There are (l-1) possible crossover points in strings of length l. In the example, a single crossover point is chosen randomly as 4 (as indicated by the symbol "|" above). Exchanging substrings around the crossover point results in two new strings for the next generation:

$$A_1' = 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0$$

$$A_2' = 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1$$

The recombination operation can take many forms.

A single point mutation:

For a binary string, random mutations alter a small percentage of the bits in the list of chromosomes. A single point mutation changes a "1" to a "0" or visa versa. Mutation points are randomly selected.

This operator alters some random value within a string. For example, take string A_1 = 1 1 0 1 1 and assume that position 3 is chosen randomly to mutate. The new string would be A_1' = 1 1 1 1 1. As in nature, the probability of applying mutation is very low in GAs, but the probability of crossover is usually high.

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, a *single point mutation* is applied with a mutation rate of μ_d. The value of μ_d used is 0.02.

4.5 Selection and Generation of Future Generations

Each individual in the population has a fitness value. In general, the fitness is a payoff measure that depends on how well the individual solves the problem. In particular, GAs are often used as optimizers, and the fitness of an individual is the value of the objective function at the point represented by the binary string or real value. Selection uses the fitness value to identify the individuals that will reproduce and mate to form the next generation.

Selection is significantly more general and less problem specific than evaluation. The only requirement for selection is that better solutions be preferred for survival and reproduction over worse solutions. There exists a variety of selection methods, the most popular of which are elitist, tournament, rank and truncation selection schemes. Each of these selection schemes is briefly described in relation to fitness maximization problems.

Elitist selection is not really a selection scheme, but a modification to other selection schemes. Elitist selection simply requires that the best individual always survives. Empirical results have shown that elitist selection schemes often outperform non-elitist selection. Thus, it is not uncommon to see the addition of explicit elitism into tournament and proportional selection schemes. The selection process is based on the survival of the fittest. Elitism keeps the best chromosomes from generation to generation. The rest die off.

This process of elitism must occur at each iteration of the algorithm to allow these two populations of joined chromosomes to evolve over the generations to the most fit members as defined by the fitness function. It decides which chromosomes are fitted enough to survive and possibly reproduce offspring in the next generation. Two phases of sorting and ranking are applied to ensure that the elitism selection is based on the population's highest fitness where fitness is defined as the ability to survive and reproduce in a specified environment.

For both populations of Cpop and Dpop in this application design, the generation of future generations is achieved via elitism, various ways of selection, as well as crossover and mutation. 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 12% of *Cpop* and 10% of *Dpop*. Another 12% of *Cpop* and 10% of *Dpop* are generated via the crossing over of pairs of elite individuals, to generate an equal number of children. The remaining (76% of *Cpop* and 80% of *Dpop*) 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 20% for *Cpop* and 2% for *Dpop*.

4.6 Fitness Function

The choice of a fitness function is not an easy thing and plays an important role in the success of the solving of a problem using GAs. Deciding which parameters to use to produce the desired output and fitness function are the important factors for Genetic Algorithms. An appropriate fitness function and the parameters to use are closely related. A fitness function generates an output from a chromosome (a set of input parameters). The fitness function may be a mathematical function, an experiment, or a game. The object is to modify the output in some desirable fashion by finding the appropriate values for the input chromosomes.

4.6.1 Previous Work in Dynamic Number of Clusters

The clustering was viewed as an optimization problem that tried to optimize the clustering objective function. Most of these clustering algorithms require the user to provide the number of clusters as input. But the user in general has no idea about the number of clusters. Hence, the users are forced to try different numbers of clusters when using these clustering algorithms. This is tedious and the clustering result may be no good especially when the number of clusters is large and not easy to guess[TY01].

The most common approach to finding cluster centers, equivalently quantized vectors, is the k-means clustering algorithm, which is a gradient descent type search algorithm. In addition to having the drawback of susceptibility to local optima, the k-means algorithm cannot be used to determine the number of classes. The number is assumed to be a prior known.

As the K-means algorithm for partitional clustering requires that the user specify the desired number of cluster [MB00][Lee02], a evolution strategy implementing variable length clustering in x-y plane was developed to address the problem of dynamic partitional clustering [Lee02]. As opposed to static, dynamic partitional clustering did not require the a priori specification of the number of clusters and hence cluster number gave rise to a variable complexity search.

A strategy implementing variable length genomes was applied for the application of dynamic clustering to solve 2-D spatial data clustering problems. A modification of the Mean Square Error (MSE) clustering measure was chosen as the fitness function. Since MSE can always be decreased by adding a data point as a cluster center, fitness is a monotonically decreasing function of cluster number. Obviously, the best clustering was not simply the data set (since no new information was gained if this was the case). The problem was that there needed to be a penalty for model complexity (i.e., number of clusters). This tradeoff between model complexity and goodness of fit was well known in function approximation and learning system theories.

As the fitness function (MSE) was poorly suited for comparing clustering that have a different number of clusters, in [Lee02] Lee proposed a heuristic MSE with dynamic cluster *n*, given by

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

The heuristic penalizes model complexity by multiplying the MSE fitness by a constant 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.

4.6.2 Problem Development

MSE heuristic fitness, reviewed above, was applied for dynamic clustering with constant dimensions (x-y plane). In the designs of coevolutionary GA-based dynamic clustering with feature selection, there are two dynamic variables interchanged with two populations: dynamic clustering and dynamic feature dimensions. A new type of *MSE extended fitness* is proposed in our model which measures quantities of both object tightness f_T and cluster separation f_S :

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

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

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

n: dynamic no. of clusters
k: dynamic no. of features
c_i: the ith cluster center
m_i: the number of data points belonging to the ith cluster
x_j^i: the jth data point belonging to the ith 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 MSE extended fitness are chosen to be unbiased in the dynamic coevolutionary environment. The principle of the MSE extended fitness is to optimize the distance criterion by minimizing the within cluster spread (inner's) and maximizing the inter-cluster separation (inter's).

4.7 Convergence Testing

Initially, each member of the chromosome population in a genetic algorithm is evaluated for their associated fitness (or costs). As the genetic algorithm proceeds into iteration, the average fitness of the chromosome population should begin to increase. This indicates that the surface is being successfully explored and that optimization is proceeding. They may however, come a point where repeated applications of genetic operators as well as selective pressure on the population will yield little or no effect on the fitness associated with the most elite solution in the current generation (neglecting any immediate effects from mutation). This is a termination criterion for the genetic algorithm. At this point the algorithm is said to have reached convergence. Another way to state this is that when the majority of the chromosomes in a population contain the same allele for that gene the genetic algorithm can be considered as converged.

The number of generations prior to termination depends on whether an acceptable solution is reached or a set number of iterations is exceeded. Most genetic algorithms keep track of the population statistics in the form of population 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 0.001 for 10 consecutive generations.

4.8 Implementation

When the genetic algorithm is implemented, it is usually done in a manner that involves the following cycle, evaluating the fitness of all of the individuals in the population. Then creating a new population by performing operations such as crossover, fitness-proportionate reproduction and mutation on the individuals whose fitness has just been measured. After that, discarding the old population and iterate using the new population. One iteration of this loop is referred to as a generation. The first generation (generation 0) of this process operates on a population of randomly generated individuals. From there on, the genetic operations, in concert with the fitness measure, operate to improve the population.

Deciding upon the sizes of the initial and generational populations is difficult. There is some tradeoff between population sizes and the number of generations needed to converge. Experience indicates the most effective population size is dependent on the problem being solved, the representation used, and the operators manipulating the representation [HH98].

To begin GA coevolution, we define DPopu with 500 population of binary-string chromosomes. 50 mothers and fathers pair from the top to the bottom. The remaining of 400 offspring are randomly produced with single-point crossover and a mutation rate (μ_d) of 0.02. The size of Cpopu is initialized at 88 individuals, from which 10 members were selected to produce the same number of 10 direct new copies in the next generation. The remaining of 68 populations are given randomly, using the dimension fine-tuner of crossover strategy, with a mutation rate μ_c of 0.2. Each pair of both CPopu and DPopu produces two children that contain traits from each parent. In addition, their two parents survive to be part of the next generation.

To decide which chromosomes in DPopu fitted enough to survive and possibly reproduce children in the next generation, the average of 4 chromosomes of CPopu is taken to evaluate each DPopu chromosome. On the other hand, each real-value chromosome of CPopu is evaluated by a set of features, and assigns the resulting fitness to CPopu spool. A flow chart of cooperative coevolution results is given in Figures 4.2.

As shown in Figure 4.4, 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 Figure 4.4) 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.

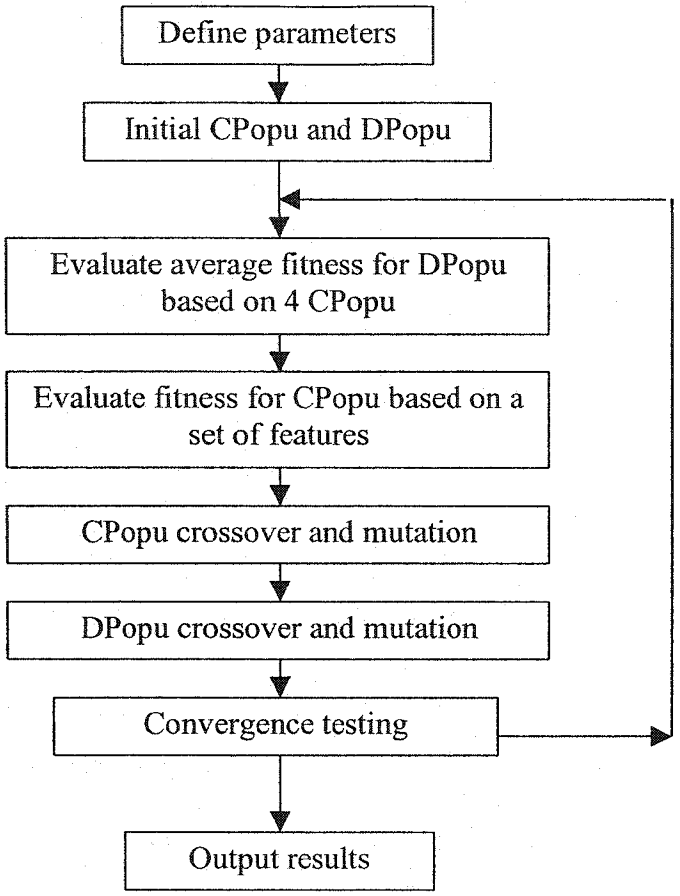
Define parameters
Initial CPopu and DPopu
Evaluate average fitness for DPopu based on 4 CPopu
Evaluate fitness for CPopu based on a set of features
CPopu crossover and mutation
DPopu crossover and mutation
Convergence testing
Output results

Figure 4.2 Diagram of cooperative coevolutions

**CPopu:*	*Cluster population*
***DPopu:*	*Dimension population*

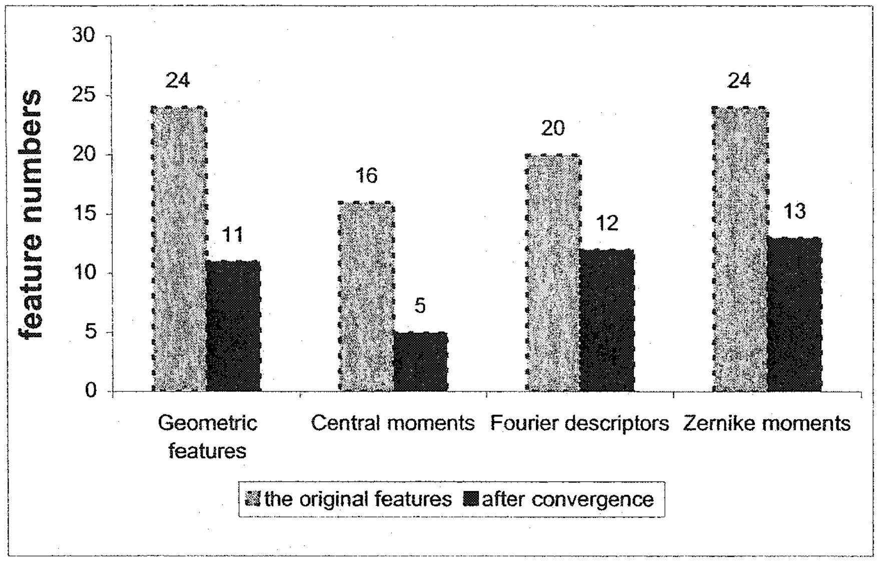

Figure 4.3 Feature distribution results

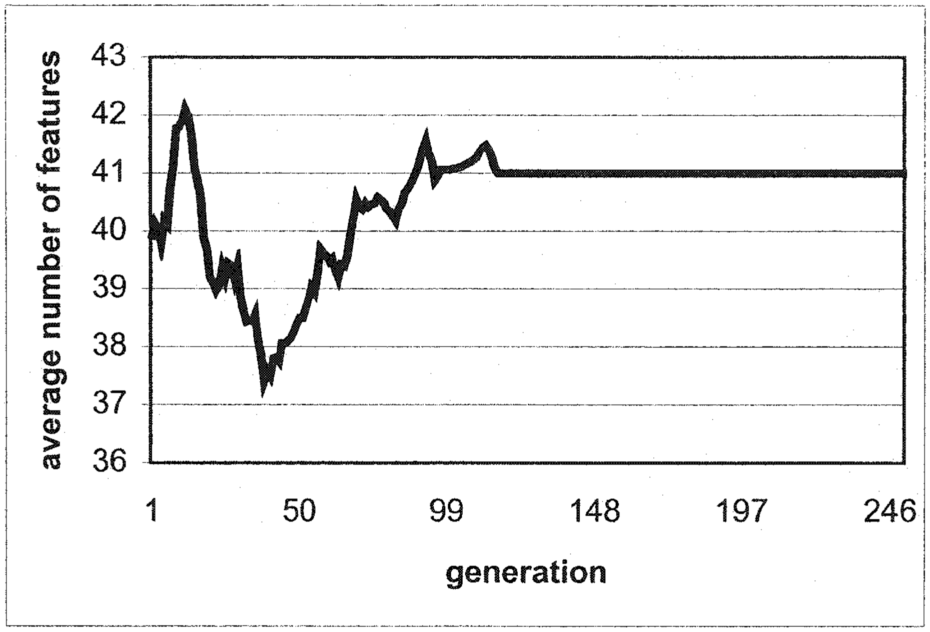

Figure 4.4 Number of feature selection after convergence

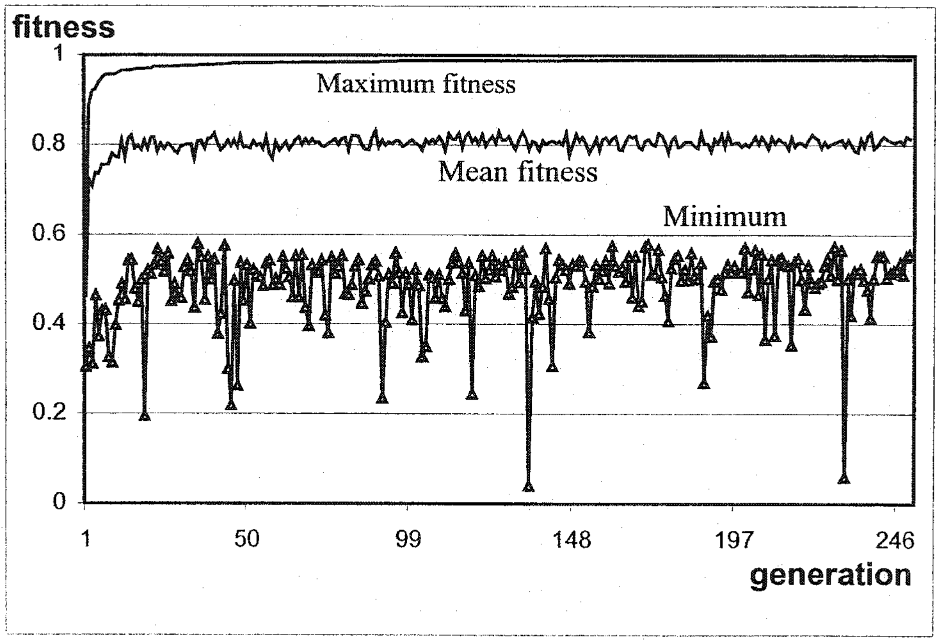

Figure 4.5 Convergence results

Chapter 5

Conclusions, Analysis, and Future Works

5.1 Results & Analysis

This thesis presents an approach for using Cooperative Co-evolutionary Clustering Algorithm to create and refine clusters of hand patterns and feature selection simultaneously with both dynamic variables of cluster centers and feature dimensions, which are divided into two population spaces.

The experiment presented here used 100 hand images and 84 normalized features. Termination occurred at a maximum of 250 generations, since it was discovered that fitness converged to less than 0.0001 variance prior. The results are promising; the average co-evolutionary clustering fitness is 0.97368 with a significantly low standard deviation of 0.1108. The average number of clusters is 4, with a very low standard deviation of 0.4714. The average misplacement number of hand image is 5.8 out of 100 hand images, with the highest misplacement number of 7 out of 100 hand images and the lowest is 2 out of 100 hand images. Following convergence, the dimension of the feature space is 41, with zero standard deviation. Hence, half of the original 84 features are eliminated.

The main benefit of the thesis is that the proposed CCCA design integrates feature selection and object clustering into one single methodology. As with the proposed co-evolutionary clustering, the input features are transformed to a new smaller set of features. The analysis of output features that proves sufficient for hand image grouping can lead to a deeper understanding of the data, allowing feature selection to be used in exploratory object classification.

In addition to the main contribution of this thesis, it also offers a number of other significant results of, and tools for research.

First, empirical results (shown in Table 2.2) 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, it introduces a number of new or modified structural (e.g. palm aspect ratio) and statistical features (under the basis of finger 1D contour transformation) that is used in our clustering algorithm, and may prove equally useful to others working in Biometric-based technologies.

Third, despite its complexity, an *MSE extended fitness* function is established, which is particularly suited to an integrated dynamic clustering space.

Forth, since the proceeding of finger upright reorientation, fingers are isolating multi-contour patterns. The approach of contour 1-D transformation served for the finger shape analysis is to improve image discrimination for the finger orientation-dependent in the projections of X-Y plane.

5.2 Discussion

In pattern recognition, features are utilized to identify one class of patterns from another. The pattern space is usually of high dimensionality. The objective of feature selection is to characterize the object, and further, to reduce the dimensionality of the measurement space to a space suitable for the application of pattern classification techniques. It is not often known in advance which features will provide the best discrimination between classes, and it is usually not feasible to measure and represent all possible features of the objects being classified. As a result, feature selection has become important techniques for automated pattern recognition, exploratory data analysis, and data mining.

It is well known that the individual hand features themselves are not very descriptive; devising methods to combine these non-salient individual features to attain robust positive identification, known as hand feature selection, is a challenging pattern recognition problem in its own right. Although engineering design has always been associated with human creativity and skill, the generation of designs can be formalized in a structured and algorithmic manner. Such a formalization of design synthesis enables automatic design synthesis through computation.

In the thesis, the application of the cooperative co-evolution involves a number of important considerations. The first decision to take when applying such an algorithm is how to encode candidate solutions within the genome. The representation must allow for the encoding of all possible solutions while being sufficiently simple to be searched with in a reasonable amount of time. Next, an appropriate fitness function must be defined for evaluating the individuals and able to adapt *the dynamic world*. The (usually scalar) fitness must reflect the criteria to be optimized and their relative importance. Representation and fitness are thus clearly problem-dependent in contrast to selection, crossover, and mutation, which seems *prima facie* more problem-independent.

At the same time, GAs applied in the thesis attempt to find the optimal solution to the problem at hand by manipulating our 2 populations (Cpop and Dpop) of candidate solutions. The populations are evaluated and the best solutions are selected to reproduce and mate to form the next generation. Over a number of generations, good traits dominate the populations, resulting in an increase in the quality of the solutions.

The study of the thesis is the first to use a genetic algorithm to simultaneously achieve dimensionality reduction and object (hand image) clustering. The purpose is to establish an alternative means of hand image classification, one that does not require extensive expert intervention. The main tool for accomplishing this is a genetic algorithm (GAs). In order to do this, a cooperative co-evolutionary GA is crafted, one that uses two populations of part-solutions in order to evolve complete highly fit solutions for the whole problem. It does succeed in both its objectives. The results show that the dimensionality of the clustering space was cut in half. The number (4) and quality (0.058) of clusters produced are also very good. These results open the way towards other cooperative co-evolutionary applications, in which three or more populations are used to co-evolve solutions and designs consisting of 3 or more loosely coupled sub-solutions or modules.

5.3 Future Works

As for the future, one main challenge to a speedy proliferation of cooperative 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 simulations 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 the thesis study can be extended further via:

(a) further fine-tuning of the GA parameters – a self-tuning parameter-less GA is currently under development;

(b) use of more powerful computers that would allow us to run larger populations for longer – a Beowulf cluster has been wired up and will be used for implementing a parallelized version of the algorithm;

(c) there are an unlimited variable number of design parameters in design co-evolution problems which often have solution spaces that are of variable complexity or dimensionality. In other words, it is up to the designer and the design process to determine the appropriate numbers;

(d) defining fitness function is also a challenge. The environment of the proposed CCCA is double-dynamic. The exploration spaces of cluster population and feature dimension population are constantly changing. These changes need to act to revise more new optimization equations and might result in new model of coevolutionary strategies;

(e) An extension fitness function is made for canonical chromosomes to enable variable length search and therefore variable complexity design synthesis. The crossover methods are made explicit and real valued rather than the typically used integer values that are encoded implicitly by gene position. The work of this thesis paves the road for more applicable co-evolution design combined with

(i) integration of feature extraction and feature selection;

(ii) feature weight; and

(iii) classifer training.

This remains the most urgent task for future research.

Reference

[Cuc98] Rita Cucchiara, "Genetic algorithms for clustering in machine vision," *Machine vision and application* 11:1-6, 1998.

[Fu82] K.S. Fu, *Applications of Pattern Recognition*, CRC Press, Inc., 1982.

[Har75] J.A. Hartigan, *Clustering Algorithms*, Wiley, New York, 1975.

[HCL03] Chin-Chuan Han, Hsu-Liang Cheng, Chih-Lung Lin, Kuo-Chin Fan, "Personal authentication using palm-print features," *Pattern Recognition* 36 (2003) 371-381.

[HH98] Randy L. Haupt and Sue Ellen Haupt, *Practical Genetic Algorithms*, Wiley Interscience, New York,1998.

[Jai99] Anil K. Jain, *Biometrics Personal Identification in Networked Society*, Kluwer Academic Publishers, 1999.

[Jam88] Mike James, *Pattern Recognition*, John Wilet & Sons, New York, 1988.

[JPR98] Anil K. Jain, S.Prabhakar, and A. Ross, "Biometrics-based web access," *MSU Technical Report TR98-33*, 1998.

[JRP99] Anil K. Jain, Arun Ross and Sharath Pankanti, "A prototype hand geometry-based verification system," *2nd Int'l Conference on Audio- and Video- based Biometric Person Authentication*, Washington D.C., 1999, 1-6.

[KSG03] N. Kharma, C. Y. Suen, P. F. Guo, "PalmPrints: a novel co-evolutionary algorithm for clustering finger images," *Genetic and Evolutionary Computation Conference,* Chicago, Illinois USA, 2003, 322-331.

[KSG03~] N. Kharma, C. Y. Suen, P. F. Guo, F. Cianci, H. Patel, P. Prakash and E. Tochtamis, "PalmPrints: a cooperative co-evolutionary algorithm for clustering hand images," *IEEE Transactions On Systems, Man, and Cybernetics Society* (submitted)

[KIM02] Nobuhiro Kanaya, Youji Iiguni and Hajime Maeda, "2-D DOA estimation method using Zernike moments," *Signal processing* 82 (2002) 521-526.

[KJ00] Ludmila I. Kuncheva and Lakhmi C. Jain, "Designing classifier fusion systems by genetic algorithms," *IEEE Transactions On Evolutionary Computations*,Vol.4, No.4, Sept., (2000) 327-336

[KS00] Mineichi Kudo, Jack Sklansky, "Comparison of algorithms that select features for pattern classifiers," *Pattern Recognition* 33 (2000) 25-41.

[KW00] G.Keller and B.Warrack, *Statistics for Management and Economics*, 5th Edition, Duxbury Press, 2000

[Lee02] C.-Y. Lee, *Efficient automatic engineering design synthesis via evolutionary exploration*, Ph.D thesis, California Institute of Technology, Pasadena, California, 2002.

[LTC97] Jiming Liu, Y.Y.Tang, and Y.C.Cao, "An evolutionary autonomous agents approach to image feature extraction," *IEEE Transaction on Evolutionary Computation,* Vol. 1, No.2, July (1997) 141-158.

[MB00] Ujjwal Maulik, Sanghamitra Bandyopadhyay, "Genetic algorithm-based clustering technique," *Pattern Recognition* 33 (2000) 1455-1465.

[OLM02] I.-S.Oh, J.-S Lee and B.-R Moon, "Local search-embedded genetic algorithms for feature selection," *Proc. of International Conf. on Pattern Recognition,* (2002) 148-151.

[Par95] Jan Paredis, "Coevolutionary computation," *Artificial Life* 2:355-375 (1995).

[PS01] Carlos Andres Pena-Reyes, Moshe Sipper, "Fuzzy CoCo: a cooperative-coevolutionary approach to fuzzy modeling," *IEEE Transaction on Fuzzy Systems* Vol. 9, No.5, October (2001) 727-737.

[RPG00] Michael L. Raymer, William F. Punch, Erik D. Goodman, Leslie A. Kuhn, and Anil K. Jain, "Dimensionality reduction using genetic algorithms," *IEEE Transaction on Evolutionary Computation,* Vol. 4, No.2, July (2000) 164-171.

[TY01] Lin Yu Tseng, Shiueng Bien Yang, "A genetic approach to the automatic clustering problem," *Pattern Recognition* 34 (2001) 415-424.

[SM97] Sunil Sharma and M.S.Prasad, "Human hand recognition system," *SPIE* Vol.3074 (0277-786X) / (1997) 160-170.

[SRB01] W.Shu, G. Rong, Z. Bian and D. Zhang, "Automatic palmprint verification," *International Journal of Image and Graphics,* Vol. 1, No.1, (2001) 135-151

[SS89] W.Siedlecki and J.Sklansky, "A note on genetic algorithms for large-scale feature selection," *Pattern Recognition Letter,* Vol. 10, 335-347 (1989).

[SSG00] R. Sanchez-Reillo, C. Sanchez-Avila and A. Gonzalez-Marcos, "Short papers: biometric identification through hand geometry measurements," *IEEE Transaction on Pattern Analysis And Machine Intelligence,* Vol. 22, No.10, Oct (2000) 1168-1171

[The89] C.W. Therrien, *Decision Estimation and Classification*, John Wiley & Sons, Inc., 1989

[WBM98] 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 (1998), pp. 347-353

[YLZ02] Jane You, Wenxin Li and David Zhang, "Hierarchical palmprint identification via multiple feature extraction," *Pattern Recognition* 35 (2002) 847-859.

[Zha00] D.D. Zhang, *Automated Biometrics: Technologies and Systems*, Kluwer Academic Publishers, Dordrecht, 2000.

[Zhe00] Miao Zhenjiang, "Zernike moment-based image shape analysis and its application," *Pattern Recognition Letters* 21 (2) (2000) pp. 169-177

[ZL02] Dengsheng Zhang and Guojun Lu, "Shape-based image retrieval using generic Fourier descriptor," *Signal Processing: Image Communication* 17 (10) (2002) pp. 825-848

Appendix: Pseudo-Algorithm for Finger Upright Reorientation

```
//find the finger rotation line OmOj
∀ pixels on the fingers
FOR j = 0: 5 fingers
{
    find two phalange points A and B, connect points A and B;
    take the midpoint, named Om (Xm,Ym), between the line AB;
    find the finger midpoint center Oj;
    connect two points Om and Oj, get the finger rotation line OmOj.
}
END


//transformation from X-Y plane to X'-Y' plane
∀ pixels on the fingers
FOR j = 0: 5 fingers
{
   take  the point Om (Xm,Ym) as a new original of coordinate system in X'-Y' plane;
   transform coordinate system from X-Y plane to X'-Y' plane is:
         X' = X – Xm
         Y' = Y – Ym
}
END
```

```
//rotate at the new original of coordinate system with an rotation angle $\theta j$

$\forall$ pixels on the fingers
FOR j = 0: 5 fingers
{
  using Cosine Theorem, calculate rotation anger $\theta j$ between line $OmOj$ and Y-axis, or
  between Y'-axis and Y''-axis;
  rotate at the new original $Om$ (Xm,Ym) with the assigned rotation anger $\theta j$, using the
  Coordinate System Rotation Principle:
            $X'' = X' \cos\theta j - Y' \sin\theta j$
            $Y'' = X' \sin\theta j + Y' \cos\theta j$
}
END

$\forall$ pixels on the fingers
FOR j = 0: 5 fingers
{
  move horizontally and vertically above $Y''=0$ and isolate each along the X'' axis;
}
END
```