Coevolutionary Feature Synthesized EM Algorithm for Image Retrieval

Rui Li, Bir Bhanu and Anlei Dong
Center for Research in Intelligent Systems
University of California
Riverside, CA, 92521
1-951-827-3954

{rli, bhanu, adong}@vislab.ucr.edu

ABSTRACT

As a commonly used unsupervised learning algorithm in *Content-Based Image Retrieval* (CBIR), *Expectation-Maximization* (EM) algorithm has several limitations, especially in high dimensional feature spaces where the data are limited and the computational cost varies exponentially with the number of feature dimensions. Moreover, the convergence is guaranteed only at a local maximum. In this paper, we propose a unified framework of a novel learning approach, namely *Coevolutionary Feature Synthesized Expectation-Maximization* (CFS-EM), to achieve satisfactory learning in spite of these difficulties. The CFS-EM is a hybrid of *coevolutionary genetic programming* (CGP) and EM algorithm. The advantages of CFS-EM are: 1) it synthesizes low-dimensional features based on CGP algorithm, which yields near optimal nonlinear transformation and classification precision comparable to kernel methods such as the *support vector machine* (SVM); 2) the explicitness of feature transformation is especially suitable for image retrieval because the images can be searched in the synthesized low-dimensional space, while kernel-based methods have to make classification computation in the original high-dimensional space; 3) the unlabeled data can be boosted with the help of the class distribution learning using CGP feature synthesis approach. Experimental results show that CFS-EM outperforms pure EM and CGP alone, and is comparable to SVM in the sense of classification. It is computationally more efficient than SVM in query phase. Moreover, it has a high likelihood that it will jump out of a local maximum to provide near optimal results and a better estimation of parameters.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing – *Algorithms, Indexing methods.*

General Terms

Algorithms, Experimentation.

Keywords

Coevolutionary Feature Synthesis, Genetic Programming, Expectation Maximization algorithm, Semi-supervised Learning, Content-Based Image Retrieval

1. INTRODUCTION

High feature dimensionality is one of the major challenges in content-based image retrieval. The supervised learning (i.e., training by using labeled samples) can often overcome the "curse of dimensionality" in many pattern recognition applications, and the two representative approaches of modern supervised learning are *support vector machine* (SVM) algorithm [1] and boosting algorithm [2]. However, it is unrealistic to obtain a large amount of labeled data in CBIR due to the high cost of data collection and the subjectivity of human's image perception. For this reason, unsupervised learning like EM is also widely used in CBIR. But EM needs a large amount of data especially when feature dimension is high, which is not always available. Thus, the learning for image databases usually can only be carried on with the hybrid labeled/unlabeled data, and this is called semi-supervised learning [3][4][5].

The basic idea behind these semi-supervised learning approaches is that a feature transformation and a classifier are learned from labeled data. The unlabeled data are transformed using the learned transformation. The labeled data and the transformed unlabeled data boost each other to improve the class distribution learning. As the labeled-data-based learning always plays a key role in this process, we need to find an appropriate method to deal with the high feature dimensionality of image databases.

Fisher discriminant analysis or multiple discriminant analysis (MDA) [6] are well-known approaches to obtain discriminant features by using a linear transformation. They are based on the assumption of Gaussian distribution. The kernel trick [7] can be combined with Fisher discriminant analysis so that a nonlinear transformation is achieved [8].

In the CBIR field, a variety of interesting approaches have been proposed to deal with high-dimensional features. Swets and Weng [9] propose a self-organizing hierarchical optimal subspace learning and inference framework (SHOSLIF) for image retrieval. The main idea is to recursively implement linear discriminant analysis on the data subsets obtained from the recursive subdivision of the data, so that the limitations of the global linear transformation are overcome. Such a hierarchical linear analysis is a nonlinear approach in nature. Su et al. [10] exploit *principal component analysis* (PCA) for dimensionality reduction by using relevance feedback. Wu et al. [11] reduce the feature dimensionality for image databases using the approach of *weighted multi-dimensional scaling* (WMDS), whose main characteristic is to preserve the local topology of the high dimensional space. The *non-negative matrix factorization* (NMF) approach [12] yields a part-based representation instead of a holistic representation by using non-negativity constraints. It is found to be useful for face recognition [13] and document retrieval [14]. The boosting approach [15] is used for image retrieval as its strong classifier consists of some weak classifiers corresponding to visual image features [16][17]. He et al. [18][19] propose the approach of *locality preserving projections* (LPP) for face analysis and image retrieval. The advantage of LPP is that it preserves local information by detecting nonlinear manifold structure.

Dong and Bhanu [20] use a *coevolutionary genetic programming* approach to reduce feature dimensionality. The main idea is to generate a low dimensional synthesized feature vector from the original high dimensional feature vector. Compared with the above methods, CGP does not assume any class distribution in the original visual feature space; and it yields explicit transformation for dimensionality reduction so that the images can be searched in the low-dimensional feature space. However, this supervised method by itself is not sufficient to solve the problems associated with CBIR as mentioned earlier.

In this paper, we propose to use a small amount of labeled data to get a weak classifier using *Coevolutionary Feature Synthesis* (CFS) based on CGP and use a large amount of unlabeled data which characterizes the joint probability distribution across different features to boost the weak classifiers by exploring discriminating features in a self-supervised fashion. Thus, a trade-off between supervised and unsupervised learning is carried out. It is to be noted that as a hill-climbing algorithm, EM converges at a local maximum, which may not be the global maximum. Although not guaranteed, CGP can possibly help to jump out of the local maximum within EM iterations and potentially allow to achieve the global maximum.

The contributions of this paper include: 1) a new algorithm which combines *Coevolutionary Feature Synthesis* (CFS) with the *Expectation-Maximization* (EM) algorithm to boost unlabeled data; 2) the exploitation of CGP algorithm to reduce the feature dimensionality; 3) the experimental results on various data sets and the comparison of different approaches. They show the efficacy and efficiency of the proposed approach.

The rest of this paper is organized as follows. EM algorithm, CGP approach and the CFS-EM algorithm are described in Section 2. Extensive comparisons between CFS-EM, CGP, pure EM and SVM are presented in Section 3. Section 4 concludes the paper.

2. TECHNICAL APPROACH

In this section, the principle of EM, CGP, CFS-EM and how CFS-EM can be used in CBIR are described in detail.

2.1 Expectation-Maximization (EM)

We assume feature vectors in low dimension follow a C component *Gaussian Mixture Model* (GMM). They can be regarded as samples of a d-dimensional random variable $\mathbf{X}$, where $\mathbf{x} = [x_1, \ldots, x_d]^T$ represent a particular sample.

Its probability density function is shown in equation (1). In this definition, $\alpha_1, \ldots, \alpha_C$ are the *mixture probabilities*, so they must be positive and sum up to 1. Each θ_i is the set of parameters for the ith Gaussian component, which includes mean $\boldsymbol{\mu}_i$ and covariance matrix $\boldsymbol{\Sigma}_i$. Thus $\theta \equiv \{ \alpha_1, \ldots, \alpha_C, \theta_1, \ldots, \theta_C \}$ is the complete set of parameters needed to specify the mixture [21].

$$p(\mathbf{x}|\theta) = \sum_{i=1}^{C} \alpha_i f_i(\mathbf{x}|\theta_i) = \sum_{i=1}^{C} \alpha_i f(\mathbf{x}|\mathbf{u}_i, \boldsymbol{\Sigma}_i)$$

$$\alpha_i > 0, \; i = 1, \cdots, C, \text{ and } \sum_{i=1}^{C} \alpha_i = 1 \qquad (1)$$

$$f(\mathbf{x}|\mathbf{u}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_i|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{u}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\mathbf{u}_i)\right]$$

Given a set of N independent samples of $\mathbf{X}$: $\mathcal{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$, the log-likelihood corresponding to the mixture is:

$$\log p(\mathcal{X}|\theta) = \log \prod_{n=1}^{N} p(\mathbf{x}^{(n)}|\theta) = \sum_{n=1}^{N} \log \sum_{i=1}^{C} \alpha_i p(\mathbf{x}^{(n)}|\theta_i) \qquad (2)$$

The goal is to find θ which maximizes log $p(\mathcal{X}|\theta)$ (*maximum likelihood, ML*) or log $p(\mathcal{X}|\theta)$+log $p(\theta)$ (*maximum a posteriori, MAP*).

Expectation-Maximization (EM) is such an algorithm [22]. It is based on the interpretation of $\mathcal{X}$ as incomplete data. For Gaussian mixtures, the missing part is a set of N labels $\mathcal{Y} = [\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(N)}]$ associated with the N samples, indicating which component produced each sample. $\mathbf{y}^{(n)} = [y_1^{(n)}, \ldots, y_C^{(n)}]$, where $y_k^{(n)} = 1$ and $y_j^{(n)} = 0$ for $j \neq k$, means that sample $\mathbf{y}^{(n)}$ was produced by the kth component. The complete log-likelihood is

$$\log p(\mathcal{X}, \mathcal{Y}|\theta) = \sum_{n=1}^{N} \sum_{i=1}^{C} y_i^{(n)} \log[\alpha_i p(\mathbf{x}^{(n)}|\theta_i)] \qquad (3)$$

The EM algorithm produces a sequence of estimates $\{\hat{\theta}(t), t = 0, 1, 2, \ldots\}$ by alternatively applying the following two steps until some convergence criterion is met:

- **E-step:** Compute the conditional expectation of the complete log-likelihood, given $\mathcal{X}$ and the current estimate $\hat{\theta}(t)$. The result is the so-called Q-function:

$$Q(\theta, \hat{\theta}(t)) \equiv E[\log p(\mathcal{X}, \mathcal{Y}|\theta)|\mathcal{X}, \hat{\theta}(t)] = \log p(\mathcal{X}, Z|\theta) \qquad (4)$$

  In this equation, because of the linearity of log$p(\mathcal{X}, \mathcal{Y}|\theta)$ with respect to $\mathcal{Y}$, we only need to compute the conditional expectation $Z \equiv E[\mathcal{Y}|\mathcal{X}, \hat{\theta}(t)]$. Explicitly, they are given by equation (5), where $z_i^{(n)}$ is a posteriori probability that $y_i^{(n)} = 1$, after observing $\mathbf{x}^{(n)}$.

$$z_i^{(n)} \equiv E[y_i^{(n)}|\mathcal{X}, \hat{\theta}(t)] = \Pr[y_i^{(n)} = 1|\mathbf{x}^{(n)}, \hat{\theta}(t)] = \frac{\hat{\alpha}_i(t) p(\mathbf{x}^{(n)}|\hat{\theta}_i(t))}{\sum_{j=1}^{C} \hat{\alpha}_j(t) p(\mathbf{x}^{(n)}|\hat{\theta}_j(t))} \qquad (5)$$

- **M-step:** Update the parameter estimates according to

$$\hat{\theta}(t+1) = \arg\max_{\theta} Q(\theta, \hat{\theta}(t)) \qquad (6)$$

  in the case of ML estimation, or

$$\hat{\theta}(t+1) = \arg\max_{\theta} \left\{ Q\left(\theta, \hat{\theta}(t)\right) + \log p(\theta) \right\} \quad (7)$$

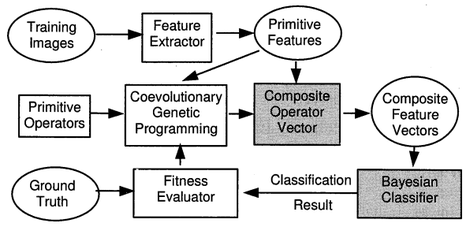
in the case of MAP estimation.

In our approach, ML estimation is used.

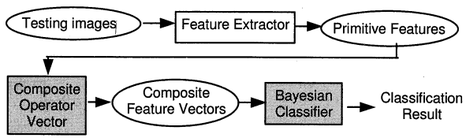
2.2 Coevolutionary Feature Synthesis

Coevolutionary feature synthesis is a *coevolutionary genetic programming* (CGP) approach. It creates low dimensional synthesized feature vectors from the original high dimensional feature vectors. The reduction is implemented by applying a series of operators on the original visual features. Such operators are called *composite operators*, which are represented by binary trees with *primitive operators* as internal nodes and original features as leaf nodes. These binary trees map the original feature space to the low-dimensional synthesized feature space, in which the images belonging to the same class form a Gaussian component no matter how these images are distributed in the original visual space. Theoretically this makes EM in low dimension possible.

The search space of all possible composite operators is so large that it is extremely difficult to find good composite operators from this vast space unless a smart search strategy is used. We follow the CGP algorithm proposed in [20], which attempts to improve the performance of object recognition. Figure 1 shows the training and testing module of the system. During training, CGP runs on labeled training images and evolves composite operators to obtain composite features. Since a Bayesian classifier is derived from the feature vectors obtained from training images, both the composite operator vector and the classifier (blocks in shade) are learned by CGP. During testing, composite feature vectors are first obtained by applying the composite operators on primitive features of the testing image, and then a Bayesian classifier is used for classification.



(a) Training module: learning composite operator vectors and a Bayesian classifier



(b) Testing module: applying learned composite operator vector and the Bayesian classifier to a test image

Figure 1. System diagram for image classification using coevolutionary genetic programming.

2.2.1 Composite Operators (Binary Tree)

The composite operators are represented by binary trees with primitive image features as the leaf nodes and primitive operators (ADD, SUB, etc.) as the internal nodes. Primitive operators will be explained in the next subsection in detail. Each tree corresponds to one composite operator, which operates on the primitive features and gives a composite feature at the root node. Figure 2 gives two sample composite operators, which are real data from our experiments. Note that not all the original features are necessarily used in feature synthesis.

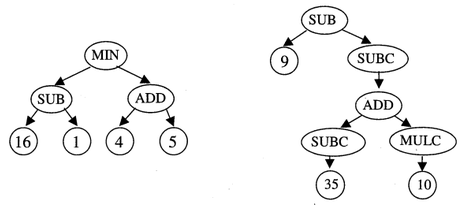


Figure 2. Sample composite operators: leaf nodes (round nodes labeled as 1, 4, 5, etc.) are original visual features and inner nodes (oval nodes labeled as ADD, SUB, etc.) are primitive operators: a simple composite operator (left figure) and a complex composite operator (right figure).

2.2.2 Primitive operator set

A primitive operator is a node in the composite operator that takes one or two real numbers, performs a simple operation on them and outputs the result. Table 1 shows the 12 primitive operators being used, where a and b are real numbers as input to an operator and c is a constant real number stored in an operator.

Table 1. Twelve primitive operators.

Primitive Operator	Description
ADD (a, b)	Add a and b.
SUB (a, b)	Subtract b from a.
MUL (a, b)	Multiply a and b.
DIV (a, b)	Divide a by b.
MAX2 (a, b)	Get the larger of a and b.
SQRT (a)	Return $\sqrt{a}$ if a ≥ 0; otherwise, return $-\sqrt{-a}$.
ADDC (a, c)	Add constant value c to a.
SUBC (a, c)	Subtract constant value c from a.
MULC(a, c)	Multiply a with constant value c.
DIVC (a, c)	Divide a by constant value c.
MIN2 (a, b)	Get the smaller of a and b.
LOG (a)	Return log(a) if a ≥ 0; otherwise, return $-\log(-a)$.

2.2.3 Fitness measure

The fitness of a composite operator vector is computed in the following way: first apply each composite operator of the composite operator vector on the original features of training images to obtain composite feature vectors of training images and then feed them to a Bayesian classifier. The classification precision of the classifier is the fitness of the composite operator vector.

2.2.4 Parameters and termination

The key parameters are the number of sub-population number S, the population size M, the number of generations G, the crossover and mutation rates, and the fitness threshold. GP stops whenever it finishes the specified number of generations or the performance of the Bayesian classifier is above the fitness threshold. After termination, CGP selects the best composite operator of each sub-population to form the learned composite operator vector to be used in testing.

2.2.5 Selection, crossover and mutation

The CGP searches through the space of composite operator vectors to generate new composite operator vectors. The search is performed by selection, crossover and mutation operations. The initial sub-populations are randomly generated. Although sub-populations are cooperatively evolved (the fitness of a composite operator in a sub-population is not solely determined by itself, but affected by the composite operators from other sub-populations), selection is performed only on composite operators within a sub-population and crossover is not allowed between two composite operators from different sub-populations.

Selection: The selection operation involves selecting composite operators from the current sub-population. In this paper, tournament selection is used and the tournament size is 5. The higher the fitness value, the more likely the composite operator is selected to survive.

Crossover: Two composite operators, called parents, are selected on the basis of their fitness values. The higher the fitness value, the more likely the composite operator is selected for crossover. One internal node in each of these two parents is randomly selected, and the two subtrees rooted at these two nodes are exchanged between the parents to generate two new composite operators, called offspring. It is easy to see that the size of one offspring (i.e., the number of nodes in the binary tree representing the offspring) may be greater than both parents if crossover is implemented in such a simple way. To prevent this *code bloat*, we specify the maximum size of a composite operator (called max-operator-size). If the size of one offspring exceeds the max-operator-size, the crossover is performed again. If the size of an offspring still exceeds the max-operator-size after the crossover is performed 10 times, GP selects two subtrees of same size (i.e., the same number of nodes) from two parents and swaps the subtrees between the parents. These two subtrees can always be found, since a leaf node can be viewed as a subtree of size 1.

Mutation: To avoid premature convergence, mutation is introduced to randomly change the structure of some composite operators to maintain the diversity of sub-populations. Candidates for mutation are randomly selected and the mutated composite operators replace the old ones in the sub-populations. There are three mutations invoked with equal probability:

- Randomly select a node of the composite operator and replace the subtree rooted at this node by another randomly generated binary tree.
- Randomly select a node of the composite operator and replace the primitive operator stored in the node with another primitive operator randomly selected from the primitive operators of the same number of input as the replaced one.
- Randomly select two subtrees of the composite operator and swap them. Of course, neither of the two subtrees can be a subtree of the other.

2.2.6 Generational Coevolutionary Genetic Programming

Generational coevolutionary genetic programming is used to evolve composite operators. The generational coevolutionary genetic programming algorithm is shown in Algorithm 1.

Algorithm 1 Generational Coevolutionary genetic programming (CGP)

Input: *primitive feature vectors, class number, primitive operator num*
Output: *composite operator vector, Bayesian classifier in low dim.*
Begin:
1. Randomly generate S sub-populations of size M and evaluate each composite operator in each sub-population individually.
2. **FOR** *gen* = 1 to G **DO**
  FOR i =1 to S **DO**
    1) Keep the best composite operator in sub-population P_i
    2) Perform crossover on the composite operators in P_i until the crossover rate is satisfied and keep all the offspring from crossover.
    3) Perform mutation on the composite operators in P_i and the offspring from crossover with the probability of mutation rate.
    4) Perform selection on P_i to select some composite operators and combine them with the composite operators from crossover to get a new sub-population P_i' of the same size as P_i.
    5) Evaluate each composite operator CO_j, $j = 1, \ldots, M$ in P_i'.
    6) Perform elitism replacement.
    7) Form the current best composite operator vector consisting of the best composite operators from corresponding sub-populations and evaluate it. If its fitness is above the fitness threshold, goto 2).
  END FOR
END FOR
3. Select the best composite operator from each sub-population to form the learned composite operator vector and output it.
End

The GP operations are applied in the order of crossover, mutation and selection. The composite operators in the initial sub-populations are randomly generated. A composite operator is generated in two steps. In the first step, the number of internal nodes of the tree representing the composite operator is randomly determined as long as this number is smaller than half of max-operator-size. Suppose the tree has p internal nodes. The tree is generated from top to bottom by a tree generation algorithm. The root node is generated first and the primitive operator stored in the root node is randomly selected. The selected primitive operator determines the number of children the root node has. If it has only one child, the algorithm is recursively invoked to generate a tree of p-1 internal nodes; if it has two children, the algorithm is recursively invoked to generate two trees of (p-1)/2 and (p-1)/2

internal nodes, respectively. In the second step, after all the internal nodes are generated, the leaf nodes containing original features are attached to those internal nodes that are temporarily the leaf nodes before the real leaf nodes are attached. The number of leaf nodes attached to an internal node is determined by the primitive operator stored in the internal node. In addition, an elitism replacement method is adopted to keep the best composite operator from generation to generation.

To evaluate CO_j for Step 5) in Algorithm 1, select the current best composite operator in each of the other sub-populations, combine CO_j with those *S-1* best composite operators to form a composite operator vector where composite operator from the *k*th sub-population occupy the *k*th position in the vector (k= *1,..., S*). Run the composite operator vector on the original features of the training images to get composite feature vectors and use them to build a Bayesian classifier. Feed the composite feature vectors into the Bayesian classifier and let the recognition rate be the fitness of the composite operator vector and the fitness of CO_j.

2.3 Coevolutionary Feature Synthesized-EM (CFS-EM)

In our research, we combine CGP with EM, to overcome the high dimensional visual feature classification task by integrating supervised and unsupervised learning paradigms and identifying most discriminating features in a self-supervised fashion. The CFS-EM algorithm is described in Algorithm 2.

Algorithm 2 Coevolutionary feature synthesized – EM (CFS-EM)

Input: *labeled training dataset L from C classes in original feature space*
Unlabeled training dataset U in original feature space
Synthesized feature dimension d
CGP parameters (population size, crossover rate, mutation rate, maximum composite operator size, etc.)
Output: *Composite operator vector* ***CO***$(\bullet)$
Bayesian classifier in low dimension θ
Begin:
Initialization:
$CO(\bullet) = CGP(L)$.
$\theta = ML(l)$.
$l = CO(L)$, $u = CO(U)$.
CFS-EM iteration:
Hill climbing:
1) Get labels (Z) for $l + u$ (calculate Z based on equation 5).
2) Modify labels (Z) for l based on ground truth.
3) Update θ based on Z.
4) If θ does not change much, goto 5), otherwise, goto 1).
Jump out of local maximum:
5) $D = L, U, + Z$.
6) $CO(\bullet) = CGP(D)$.
7) $l = CO(L)$, $u = CO(U)$.
8) Get labels for $l + u$ (calculate Z based on equation 5).
9) Update θ based on labels (Z) and $l + u$, goto 1).
end

In the initialization, the CGP is applied on *C* classes of labeled training data (L) to get a composite operator vector and a Bayesian classifier represented by the Gaussian distribution parameters θ. Both labeled training data (L) and unlabeled training data (U) are transformed into low dimension (l+u) using this composite operator vector. In CFS-EM iteration, firstly *'EM hill climbing'* is applied on this low dimension dataset to find a locally optimal Bayesian classifier. The stopping criterion for EM is that the parameters (θ) do not change for two consecutive iterations. At this time, both L and U are 'labeled' by the Bayesian classifier. In the *'Jump out of local maximum'* step, CGP can be applied on the 'labeled' whole dataset to find a better composite operator vector. Then the Bayesian classifier can be updated. The *'Hill climbing'* and *'Jump out of local maximum'* steps iterate until a certain number of iterations or a satisfactory classification performance is reached.

Figure 3 shows a conceptual illustration for hill climbing/jumping out of local maximum procedure. The mountain depicts the classification performance. The higher the position is, the better the performance is. First CGP find a classifier, then EM makes the classifier better by climbing to a local peak. In the next iteration, CGP finds a new classifier which helps it jump out of local peak so that EM can perform hill climbing again. This procedure lasts until global peak is reached or an acceptable performance is achieved. (a) shows the ideal situation, where CGP jumps higher than EM. In reality, it does not always jump higher. When it jumps out of the local maximum, it may go down to the valley, as illustrated in (b). But after enough iteration, the global peak can be possibly reached.

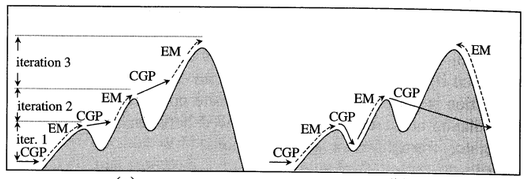


Figure 3. Conceptual illustration for hill climbing/jumping out of local maximum iterations by CGP/EM hybrid.

The advantages of CFS-EM are: 1) the unlabeled data can be boosted with the help of the class distribution learning using CGP feature synthesis approach; 2) it synthesizes low-dimensional features based on *coevolutionary genetic programming* (CGP) algorithm, which yields optimal nonlinear transformation and achieves classification performance comparable to kernel methods such as *support vector machine* (SVM); 3) the explicitness of feature transformation is especially suitable for image retrieval because the images can be searched in the synthesized low-dimensional space, while kernel-based methods have to make classification computation in the original high-dimensional space using all the labeled training data.

2.4 CFS-EM for image retrieval

CFS-EM is a general classification algorithm. In this paper, we apply it to CBIR application to demonstrate its efficacy. The system diagram is shown in Figure 4.

In training, the inputs are labeled and unlabeled data. By applying CFS-EM algorithm, best composite operator vector CO$(\bullet)$ and Bayesian classifier in low dimension (two blocks in shade) are obtained. In testing, a query image is transformed into a low dimension synthesized feature vector by applying the composite operator vector. Then this synthesized feature vector is classified using the Bayesian classifier. The images that are the nearest neighbors are returned as the retrieval result.

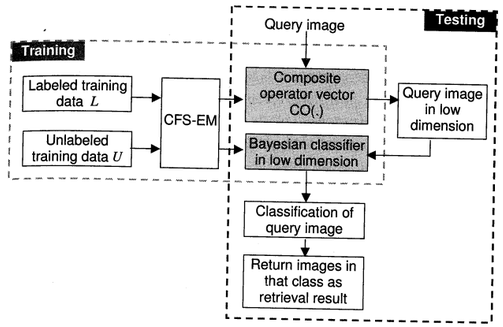


Figure 4. System for applying CFS-EM to CBIR.

3. EXPERIMENTAL RESULTS

To evaluate the efficacy of CFS-EM approach for classification, we implemented the CFS-EM algorithm on a synthetic database and two image databases. We compared our classifier with pure EM, CGP approach [20] and SVM. SVM is commonly known as the best classifier. It has shown better performance for various imaging applications. So comparison with SVM is a reasonable choice.

The programs are written in C++. They are compiled using g++ 2.95 and run on a Sun Microsystems sun4u with 2048MB memory. The operating system is Solaris 2.8.

3.1 Datasets

For experimental purposes, our databases are all labeled. So we divide the whole database into halves. One half is for training and the other half is for testing. Within training data, part of the data are defined as labeled (L) and the rest of the data are defined as unlabeled (U), as illustrated in Figure 5.

The performance of the classifier is evaluated for three databases: synthetic database *Syn-3000*, real image databases *Corel-1200* and *Corel-1500*. Both *Corel-1200* and *Corel-1500* are selected from Corel Stock Photo Library [23] that contains 200 CDs corresponding to 200 classes and each CD has 100 images.

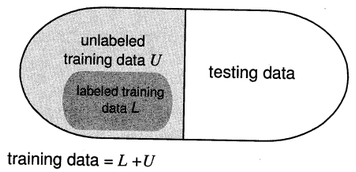


Figure 5. Data assignment for training/testing.

3.1.1 Syn-3000

3 classes of 5 dimensional multivariate Gaussian distribution are created manually. 1000 samples for each class are generated as the feature vectors. The means and covariance matrices are shown below:

$$\mu_1 = [0\ \ 0\ \ 0\ \ 0\ \ 0] \quad \mu_2 = [-1\ \ 0\ \ 0\ \ 0\ \ 0] \quad \mu_3 = [1\ \ 0\ \ 0\ \ 0\ \ 0]$$

$$\Sigma_1 = \begin{bmatrix} 1 & 3 & 0.5 & 0.7 & 1 \\ 3 & 1 & 0.8 & 0.2 & 0 \\ 0.5 & 0.8 & 0.5 & 2 & 1 \\ 0.7 & 0.2 & 2 & 1 & 0.5 \\ 1 & 0 & 1 & 0.5 & 1 \end{bmatrix}$$

$$\Sigma_2 = \begin{bmatrix} 0.5 & 2 & 0.5 & 1 & 2 \\ 2 & 0.5 & 0.3 & 1 & 0.2 \\ 0.5 & 0.3 & 3 & 0.1 & 0.8 \\ 1 & 1 & 0.1 & 0.2 & 0.4 \\ 2 & 0.2 & 0.8 & 0.4 & 5 \end{bmatrix}$$

$$\Sigma_3 = \begin{bmatrix} 1 & 2 & 0 & 0.5 & 2 \\ 2 & 0.4 & 1 & 0.6 & 0.1 \\ 0 & 1 & 0.6 & 2 & 1 \\ 0.5 & 0.6 & 2 & 1 & 0.8 \\ 2 & 0.1 & 1 & 0.8 & 3 \end{bmatrix}$$

3.1.2 Corel-1200

We select 1200 images belonging to 12 classes, and the images in each class have similar visual features, i.e., each class in the feature space forms a cluster.

Figure 6 shows sample images for each class. These 12 classes are corresponding to the CDs (series number) in the library including *Mayan & Aztec Ruins* (33000), *horses* (113000), *owls* (75000), *sunrises & sunsets* (1000), *North American wildflowers* (127000), *ski scenes* (61000, 62000), *coasts* (5000), *auto racing* (21000), *firework photography* (73000), *divers & diving* (156000), *land of the Pyramids* (161000) and *lions* (105000).

Figure 6. Sample Image of the 12 classes in the database obtained from Corel stock photo library.

Images are represented by texture features, color features and structural features. The16 texture features aremeans and standard deviations derived from 8 Gabor filters (2 scales and 4 orientations) [24]. We also extract means and standard deviations from the three channels in HSV color space. For structural features, we use the water-filling approach [25] to extract 18 features from each image. Thus, each image is represented by 40 visual features. The features are normalized to 0-100 before the experiments.

3.1.3 Corel-1500

We add 300 images (from other three CDs in the Library) into *Corel-1200* to obtain *Corel-1500*. The three new CDs (series number) are *hawks and falcons* (70000), *tigers* (108000) and *tulips* (258000). Each of these three CDs is merged to one existing CDs in *Corel-1200* to form a class, so that *Corel-1500* still has 12 classes. In these 12 classes, there are three classes each of which consists of two clusters in visual feature space: the CD of *hawks and falcons* and the CD of *owls* form the class of *bird*, the class of *tulips* and the class of *North American wildflowers* form the class

of *flowers*, and the class of *tigers* and the class of *lions* form the concept of *wild beasts*.

Figure 7 shows the sample images of these three classes containing multiple CDs. Obviously, *Corel-1500* is challenging for pure EM and linear transformation approach such as *multiple discriminant analysis* (MDA). The Same 40 features as in *Corel-1200* are extracted for this database.

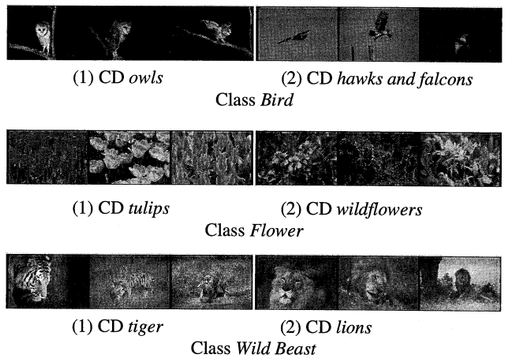


Figure 7. Corel-1500: sample images from three classes containing multiple CDs.

3.2 Classification Performance Comparisons

3.2.1 Syn-3000:

Classification precision is compared among pure EM, CGP, CFS EM and SVM (linear kernel function, *Raidial Basis Function* (RBF) kernel function and polynomial kernel functions) on *Syn-3000*. Pure EM uses parameters estimated from labeled training data (*L*) as the initial condition. CFS-EM is trained based on *L* and *U*. As supervised learning scheme, CGP and SVM use *L* for training.

For CFS-EM, there are 10 iterations for each run and the performance of the best iteration is picked as the performance of that run. Then we run the algorithm for 10 times and average the performances to describe the average behavior of the classifier. For CGP, we run it 10 times and average the performance as well.

Both CGP and CFS-EM uses the same GP parameters: (a) sub-population size: 50; (b) crossover rate: 0.6; (c) number of generation: 50; (d) mutation rate: 0.05; (e) fitness threshold: 1.0; (f) tournament size: 5; (g) composite operator num: 2. There has been a lot of discussion about how to choose GP/GA parameter during the past decades. The set of parameters used in this paper is empirically decided based on the database size, number of classes, etc [26][27]. Here we reduce the feature dimensionality from 5 to 2. Note that SVM runs on 5 dimensional data vs. 2 dimensions for CFS-EM.

Figure 8 shows the classification performance of each classifier with regard to the percentage of labeled training data. For CFS-EM and CGP, each point depicts the average performance of 10 runs. As a CGP/EM hybrid, CFS-EM is better than CGP and pure EM alone. CFS-EM has almost the same performance as linear SVM. But when kernel function changes, SVM gives much worse performance (see RBF SVM and polynomial SVM). This demonstrates how the performance of SVM depends on the kernel function. However, CFS-EM does not have this problem. This figure also shows all the six classifier give higher classification precision when percentage of labeled data increases. Especially for CFS-EM, the performance improves ~12% when the labeled data percentage increase from 5% to 10% and only increase another 2-3% when labeled data increase to 95%. This means if the users do not care about the 2-3% difference in performance, 10% labeled data are enough, which could save a lot labeling work!

Figure 9 shows the distribution of all samples in low dimension (2D) after applying the composite operators from CFS-EM. It can be seen that three classes are well separated.

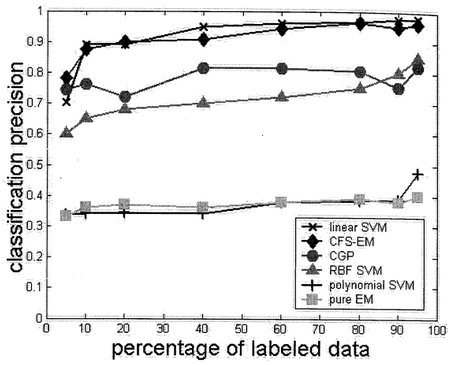


Figure 8. Classification precision comparison for *Syn-3000*.

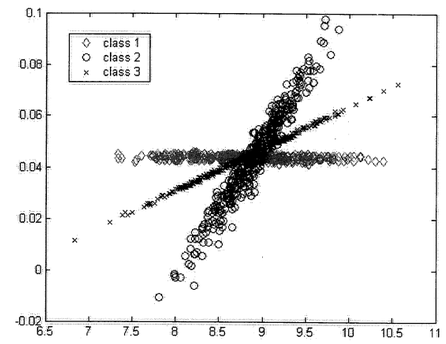


Figure 9. Distribution of three classes in 2D using CFS-EM

3.2.2 Corel-1200:

Classification performance is compared among CGP, CFS-EM and SVM on *Corel-1200* database. Pure EM is not used here because each class has only around 50 images for training, which are not enough for estimating a 40 dimensional Gaussian distribution. Both CGP and CFS-EM uses the same GP parameters: (a) sub-population size: 50; (b) crossover rate: 0.6; (c) number of generation: 10; (d) mutation rate: 0.05; (e) fitness threshold: 1.0; (f) tournament size: 5; (g) composite operator num: 6. We reduce the feature dimensionality from 40 to 6.

Figure 10 shows the classification precision comparison. It shows that when percentage of labeled data increases, performances of all the five methods increase. CFS-EM is higher than CGP. The performance of CFS-EM is comparable to linear SVM. Similar to Figure 8, RBF and polynomial SVM are much worse than CFS-EM. Since GP is slow, we only run 10 generations (50 for *Syn-3000*). If more GP generations are run, it is reasonable to believe CFS-EM will perform better. But more important than a small difference in the classification error, CFS-EM gives an explicit transformation, while SVM does not. This makes retrieval more efficient. This has been verified in this experiment. Linear SVM test needs ~5 seconds but CFS-EM only needs less than 1 second. Note that SVM runs on 40 dimensional feature vectors vs. 6 dimensions for CFS-EM. This advantage in computational cost will be addressed later in subsection 3.4.

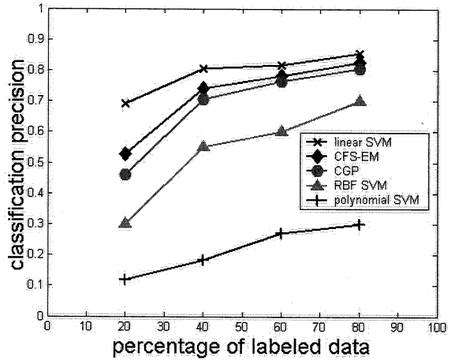


Figure 10. Classification precision comparison for *Corel-1200*.

3.2.3 Corel-1500:

Classification performance is compared among CGP, CFS-EM and SVM on *Corel-1500* database. Same trend as Corel-1200 is observed (shown in Figure 11), except the overall classification precision is little lower. SVM runs on 40 dimensional feature vectors vs. 6 dimensions for CFS-EM.

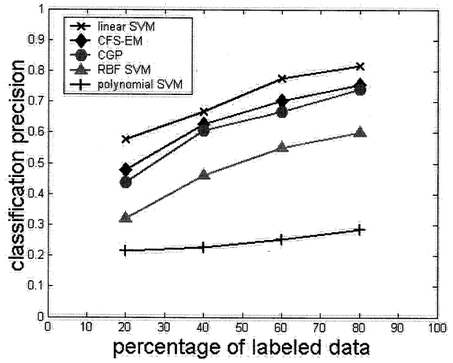


Figure 11. Classification precision comparison for *Corel-1500*.

3.3 Hill climbing/CGP process

As stated in Section 2.3, in CFS-EM, CGP helps EM to jump out of the local maximum. Figure 12 shows a real example on *Syn-3000* dataset (*L*: 40%, *U*: 60%). X-axis is the iteration number, from 1 to 10. Y-axis is the classification precision on training data (*L*+*U*). Blue bars show CGP performance and magenta bars show EM performance. This figure demonstrates the performance improvement: in 10 iterations the performance increases gradually from 84% to 94%. In iteration 1, CGP gives an initial condition based on labeled training data *L* (first blue bar), then when bringing in unlabeled training data *U*, EM hill 'climbs' and reaches a local peak (indicated by magenta bars). EM does not actually 'climb' in the first iteration, since more unlabeled training data than labeled training data (60% vs. 40%) are brought in and they might have different distribution than *L*, so the overall performance degrades. However, in the next iteration, CGP helps it jump out of the local maximum and EM does hill climb. This scheme does not guarantee a better performance than the previous iteration as shown in the conceptual illustration in Figure 3 (b), e.g., iteration 4 has almost the same behavior as iteration 3. But it is getting better in the long-term. Classification precision increases about 10% after 10 iterations.

In this iterative process, CGP serves in both a feature synthesis role and in a perturbation role to make EM in high dimension possible and allow it jump out of a local maxima to pursue the global maximum.

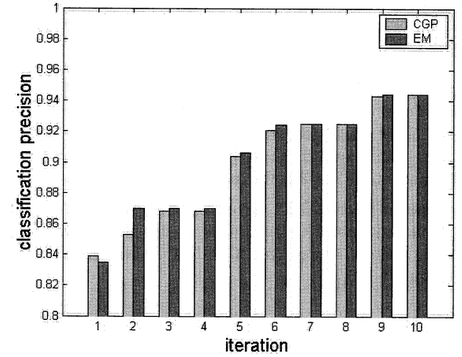


Figure 12. Classification precision for each iteration describing hill climbing/CGP process.

3.4 Computational cost: CFS_EM vs. Linear SVM

In image retrieval, query time is of great concern since users will not wait for minutes to get an answer. Compared with linear SVM, the gain in computational cost during query phase of our CFS-EM algorithm is large especially when original feature dimension is high. The parameters that affect query time for CFS-EM and linear SVM are:

- C – class number
- D – original feature dimension
- d – composite feature dimension
- T – amount of labeled training data
- α – percentage of support vectors in labeled training data

K – maximum number of nodes for composite operator

For CFS-EM, the maximum computational cost of one query is shown in equation (8). The first term is for composite operator transformation. Given K as the maximum number of nodes in a tree, the maximum computation is K when all the primitive operators have only one operand and all the K nodes are used. d composite operators gives the computational cost as $O(K \cdot d)$. Calculation of the Gaussian probability for C classes requires $O(C \cdot d^2)$. To find the nearest class we need to find the largest probability among C classes, which has the computational cost of $O(C)$. Thus, the total cost is:

$$\underbrace{O(K \cdot d)}_{\text{transformation}} + \underbrace{O(C \cdot d^2)}_{\text{Bayesian classification}} + \underbrace{O(C)}_{\text{find the closest class}} \tag{8}$$

For linear SVM, the classification of x is defined in equation (9), where x_i, $i = 1,\ldots, \alpha T$ are the support vectors, $\langle \cdot \rangle$ is the inner product and α_i, $i = 1,\ldots, \alpha T$ and b are the parameters learned from SVM training. From this definition, the computational cost is $O(D \cdot \alpha T)$.

$$f(x) = sign\left\{\sum_{i=1}^{\alpha T} \alpha_i \langle x_i, x \rangle + b\right\} \tag{9}$$

Figure 13 shows the computational cost comparison between CFS-EM and linear SVM with reference to the percentage of labeled training data for *Corel-1200*. The dotted line is the maximum cost of CFS-EM and the solid line is the cost of linear SVM. It can be seen that linear SVM cost varies linearly with percentage of labeled training data while the cost of CFS-EM does not. Other kernel functions needs even more computation than linear SVM. The figure shows that even when there are only 5% labeled training data used for linear SVM, it needs more computation than CFS-EM in the query phase.

As mentioned in Section 3.2.2, in real experiment on *Corel-1200*, the total query time of 600 images is ~5 seconds for linear SVM while less than 1 second for CFS-EM, which verifies our computation.

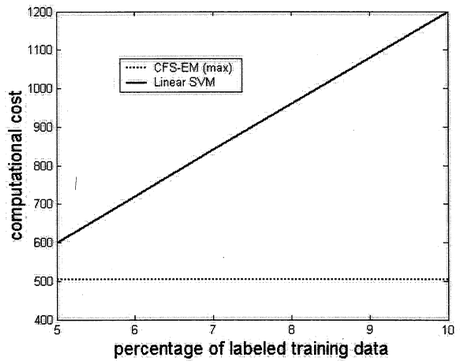


Figure 13. Computational cost: CFS-EM vs. Linear SVM. (For *Corel-1200* DB, $C = 12$, $D = 40$, $d = 6$, $T = 600*(5\%\sim10\%)$, $\alpha = 0.5$, $K = 10$)

When the percentage of labeled data is fixed, computational cost of linear SVM is linear in the dimension of original feature vector while it is fixed for CFS-EM. From Figure 14 we can see when the dimension of original feature vector is larger than 10, CFS-EM is faster than linear SVM. In both *Corel-1200* and *Corel-1500*, the original feature dimension is 40. It is believed that in most cases in practice, original dimension will be more than 10.

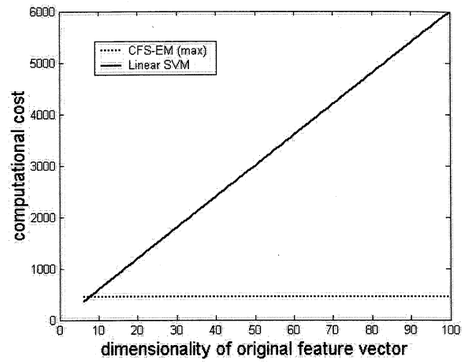


Figure 14. Computational cost: CFS-EM vs. Linear SVM. (For *Corel-1200* DB, $C = 12$, $D = 6$ ~100, $d = 6$, $T = 600*20\%$, $\alpha = 0.5$, $K = 10$)

3.5 Discussions

From experiments on both synthetic data and real image data, we find that CFS-EM gives better classification performance than CGP and EM alone. CFS-EM initializes from CGP result and hill climbs for more than once, while pure EM starts randomly and only hill climbs once. So CFS-EM is more likely to perform better than CGP and pure EM alone. In the comparison between CFS-EM and SVM, the performance of linear SVM is a couple of percentage points higher than CFS-EM, but the performance of RBF SVM and Polynomial SVM are much less. This reveals the key problem of kernel based methods: performance largely depends on the kernel function. However CFS-EM has no such limit. The small difference in performance between CFS-EM and linear SVM can possibly be reduced by running CGP for a larger number of generations within CFS-EM. Additional experiments show that CFS-EM also outperforms a combination of MDA and EM: D-EM [5]. For the computational cost comparison, since CFS-EM generates an explicit transformation from high dimension to low dimension, CFS-EM is far more efficient than SVM in the query phase. Moreover, CFS-EM can effectively solve the local maximum problem by integrating CGP with EM. Therefore, for real-time image retrieval, considering classification precision, computational time, and the convergence behavior CFS-EM outperforms SVM.

CFS-EM has some limitations. Firstly, during training it needs more learning time than SVM. When the database is not updated often, the long offline training time is not a concern. Various ideas based on incremental learning can be used to make the update more efficient, Secondly, the class number C is obtained from the labeled training data and is used for the entire learning procedure. The labeled training data must represent the whole

training data otherwise the classification performance is degraded [5]. There has been some work on how to estimate the number of classes for unlabeled data [28]. We plan to overcome these limitations of transductive learning in our future work.

4. CONCLUSIONS

Handling large dimensional feature vectors in CBIR has been challenging, especially in the statistical modeling of the database as a mixture of Gaussian. Unlike other methods, in our approach, we use a transductive learning algorithm. The proposed CGP/EM hybrid algorithm, called coevolutionary feature synthesized EM (CFS-EM) algorithm approaches this problem in the EM framework. In CFS-EM, the Gaussian class distribution in visual feature space is unnecessary. This is because it tries to form a Gaussian component for the images that belong to the same class in the low-dimensional synthesized feature space no matter how these images are distributed in the original visual feature space. By integrating CGP with EM, it overcomes the local maximum problem and it provides a better performance than by EM or CGP alone. Better than kernel based classifier like SVM, CFS-EM does not need to specify a kernel function, and it gives an explicit transformation from high dimensional visual features to low dimensional synthesized features, which makes retrieval in CBIR more efficient. As a general classification method, CFS-EM is not limited to CBIR. It can be extended to retrieve other media types involved in engineering and computer science.

5. REFERENCES

[1] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[2] R. Meir and G. Rätsch, "An introduction to boosting and leveraging," *Advanced Lectures on Machine Learning, LNCS*, pp. 119-184. Springer, 2003.

[3] S. Tong and E. Chang, "Support vector machine active learning for image retrieval," *ACM Multimedia*, pp. 107-118, 2001.

[4] T. Hertz, N. Shental, A. Bar-Hillel and D. Weinshall, "Enhancing image and video retrieval: learning via equivalence constraint," *IEEE Conf. on CVPR*, pp. 668-674, 2003.

[5] Q. Tian, J. Yu, Q. Xue and N. Sebe, "A new analysis of the value of unlabeled data in semi-supervised learning for image retrieval," *ICME*, 2004.

[6] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, John Wiley & Sons, second edition, 2001.

[7] B. SchÄolkopf and A.J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond,* the MIT Press, 2001.

[8] G. Baudat and F. Anouar, "Generlized discriminant analysis using a kernel approach," *Neural Computation,* vol. 12, pp. 2385-2404, 2000.

[9] D. Swets and J. Weng, "Hierarchical discriminant analysis for image retrieval," *IEEE Trans. PAMI*, vol. 21, no. 5, pp. 386-401, May 1999.

[10] Z. Su, S. Li, and H. J. Zhang, "Extraction of feature subspaces for content-based retrieval using relevance feedback," *Proc. ACM Int. Conf. Multimedia*, pp. 98-106, 2001.

[11] P. Wu, B. S. Manjunath, and H. D. Shin, "Dimensionality reduction for image search and retrieval," *Proc. Int Conf. Image Processing*, 2000.

[12] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[13] S. Z. Li, X.W. Hou, H.J. Zhang and Q.S. Cheng, "Learning spatially localized, parts-based representation," *IEEE Conf. on CVPR*, pp. 207-212, 2001.

[14] W. Xu, X. Liu and Y. Gong, "Document clustering based on non-negative matrix factorization," *ACM SIGIR*, pp. 267-273, 2004.

[15] R. Meir and G. Rätsch, "An introduction to boosting and leveraging," *Advanced Lectures on Machine Learning, LNCS*, pp. 119-184. Springer, 2003.

[16] K. Tieu and P. Viola, "Boosting Image Retrieval," *International Journal of Computer Vision*, vol. 56, no.1-2, pp.17-36, January-February 2004.

[17] A. Torralba, K.P. Murphy and W.T. Freeman, "Sharing features: efficient boosting procedures for multiclass," *IEEE Conf. on CVPR* pp. 762-769, 2004.

[18] X. He, S. Yan, Y. Hu, and H.J. Zhang, "Learning a locality preserving subspace for visual recognition," *ICCV*, pp.385-393, 2003.

[19] X. He, W.Y. Ma, and H.J. Zhang, "Learning an image manifold for retrieval," *ACM Conference on Multimedia*, pp.17-23, Oct 10-16, 2004.

[20] A. Dong, B. Bhanu, Y. Lin, "Evolutionary feature synthesis for image databases," *IEEE Workshop on Applications of Computer Vision*, pp.330-335, 2005.

[21] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood Estimation from Incomplete Data Via the EM algorithm," *J. Royal Statistical Soc. B,* vol.39, pp.1-38, 1977.

[22] G. McLachlan and D. Peel, *Finite Mixture Models*. New York John Wiley & Sons, 1997.

[23] http://www.corel.com, 2005

[24] B. S. Manjunath and W.Y. Ma, "Texture feature for browsing and retrieval of image data," *IEEE Trans. PAMI,* vol. 18, no.8, pp. 837-842, August 1996.

[25] X.S. Zhou, Y. Rui, and T.S. Huang, *Exploration of Visual Data,* Kluwer Academic Publishers, 2003.

[26] J.J. Grefenstette, "Optimization of control parameters for genetic algorithms", *IEEE Trans. Systems, Man, and Cybernetics,* Vol. 16-1, pp. 122-128, 1986

[27] A.E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms", *IEEE Transactions on Evolutionary Computation,* 3(2): 124-141, 1999.

[28] M.A. Figueriedo and A. Jain, Unsupervised Learning of Finite Mixture Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2002. 24(3): p. 381-396.

ACM Multimedia 2005

Proceedings of the 13th ACM International Conference on Multimedia

November 6-11, 2005 • Singapore

Sponsored by the ACM Special Interest Groups SIGMM & SIGGRAPH