The VLDB Journal (2001) 9: 279–293 / Digital Object Identifier (DOI) 10.1007/s007780100028

Combining multi-visual features for efficient indexing in a large image database

Anne H.H. Ngu[1,*], **Quan Z. Sheng**[1], **Du Q. Huynh**[2], **Ron Lei**[1]

[1] School of Computer Science and Engineering, The University of New South Wales, Sydney 2052 NSW, Australia; E-mail: anne@cse.unsw.edu.au
[2] School of Information Technology, Murdoch University, Perth 6150 WA, Australia; E-mail: d.huynh@murdoch.edu.au

Edited by T. Özsu and S. Christodoulakis. Received 11 June 1998 / Accepted 25 July 2000
Published online: 13 February 2001 – © Springer-Verlag 2001

Abstract. The optimized distance-based access methods currently available for multidimensional indexing in multimedia databases have been developed based on two major assumptions: a suitable distance function is known a priori and the dimensionality of the image features is low. It is not trivial to define a distance function that best mimics human visual perception regarding image similarity measurements. Reducing high-dimensional features in images using the popular principle component analysis (PCA) might not always be possible due to the non-linear correlations that may be present in the feature vectors. We propose in this paper a fast and robust hybrid method for non-linear dimensions reduction of composite image features for indexing in large image database. This method incorporates both the PCA and non-linear neural network techniques to reduce the dimensions of feature vectors so that an optimized access method can be applied. To incorporate human visual perception into our system, we also conducted experiments that involved a number of subjects classifying images into different classes for neural network training. We demonstrate that not only can our neural network system reduce the dimensions of the feature vectors, but that the reduced dimensional feature vectors can also be mapped to an optimized access method for fast and accurate indexing.

Key words: Image retrieval – High-dimensional indexing – Neural network

1 Introduction

Currently, intelligent image retrieval systems are mostly similarity-based. The idea of indexing an image database is to extract the features (usually in the form of a vector) from each image in the database and then to map features into points in a multi-dimensional feature space. The distance between two feature points is frequently used as a measure of similarity between the two corresponding images. Once the distance or similarity function is defined for the multidimensional feature space, a nearest neighbour search can be used to retrieve the images that satisfy the criteria specified in a given query.

[*] Currently working for Telcordia Austin Research Center, Texas, USA

Correspondence to: Anne H.H. Ngu, 10901, Spicewood Parkway, Austin TX 78750, USA

The indexing methods that have been proposed to support this kind of retrieval are known as spatial access methods (SAMs) and *metric trees*. The former includes SS-tree [31], R^+-tree [26], and grid files [11]; the latter includes the vp-tree [4], mvp-tree [1], $GNAT$ [2], and M-tree [6]. While these methods are effective in some specialized image database applications, many open problems in indexing still remain.

First, image feature vectors usually have high dimensions (e.g., some image feature vectors can have up to 100 dimensions). Since the existing access methods have an exponential time and space complexity as the number of dimensions increases, for indexing high dimensional vectors, they are no better than sequential scanning of the database. This is the well-known "dimensional curse" problem. For instance, methods based on R-trees can be efficient if the fan-out of the R-tree nodes remain greater than 2 and the number of dimensions is under 5. The search time with linear quadtrees is proportional to the size of the hypersurface of the query region which grows with the number of dimensions. With grid files, the search time depends on the directory, whose size also grows with the number of dimensions [11].

Second, one of the main differences between an image retrieval system and a traditional database system is the former's ability to rank-order results of retrieval by the degree of similarity with the query image [15]. Given a set of different feature vector types $\{\phi_1, \phi_2, \ldots, \phi_M\}$ where each set ϕ_i, for $i = 1 \ldots M$, contains feature vectors of the same number of dimensions, i.e., $\phi_i = \{\mathbf{p}_{ik} \,|\, k = 1 \ldots N_i\}$. Then a similarity function must be determined for each feature vector type. That is, we must have $\{S_i \,|\, i = 1 \ldots M\}$, where each S_i is a similarity function. When a query feature vector $\mathbf{q}$ is posed to the image database, a number of feature vectors from each set ϕ_i that satisfy a similarity criterion τ are retrieved. Consequently, a separate indexing structure is required to support retrieval based on each feature vector type.

Building a separate indexing structure for each feature type, such as colour, texture, or shape, cannot efficiently support queries that involve composite features (features of more than one type, e.g., features that are composed of both colour and texture information). To answer a query that involves a composite feature vector, a hierarchical approach is often adopted in which each component of the query is applied against an appropriate index in a lower layer. The results are then merged and presented to the user at a higher layer. For example, a query such as “find an object that is red in colour, round in shape, and has a fabric texture” can only be answered by first consulting the colour index, the shape index, the texture index, and finally returning the intersection of the three resulting sets. This is inefficient in terms of storage utilization and system performance. Furthermore, it is assumed that in a complex scene, each type of visual feature contributes equally to the recognition of that image. This phenomenon is not supported in human visual perception.

Although many research works have claimed to support queries on composite features by combining different features into an integrated index structure, very few of them explain how the integration is implemented. There are two main problems that need to be addressed here. The first one is that the integrated features (or composite features) typically generate very high dimensional vectors, which cannot be handled efficiently by the existing access methods. The other problem is the definition of image similarity measurements which reflects human visual perception. For example, in what form should the similarity function for composite features be when the contribution of each feature type is weighted differently in human visual perception?

There are two approaches to solving the indexing problem. The first approach is to develop a new spatial index method which can handle data of any dimensions and employ a k-nearest neighbourhood (k-NN) search. The second approach is to map the high-dimensional feature space into a lower dimensional feature space so that an existing access method can be applied. Creating a generalized high-dimensional index that can handle hundreds of dimensions is still an unsolved problem to date. The second approach is clearly more practical. In this work, we focus on how to reduce the dimensions of composite feature vectors so that effective index structures can be created.

The second problem is associated with human visual perception. The various visual features in an image are not weighted equally in human visual perception. In other words, the human visual system has different responses to colour, texture, and shape information in an image. When these visual features are represented by the feature vectors extracted from an image, the similarity measure for each feature type between the query image and an image in the database is typically computed by a Euclidean distance function. The similarity measure between the two images is then expressed as a linear combination of the similarity measures of all the feature types. The question that remains here is whether a *linear combination* of the similarity measures of all the feature types best reflects how we perceive images as similar. So far, no experiments have been conducted that demonstrate (or counter-demonstrate) the above belief.

The main contribution of this work is in building an effective content-based retrieval system which can efficiently support queries on composite features without the need to construct a separate indexing structure for each feature type. The core of the work is to use a hybrid method that incorporates the PCA and neural network to reduce high-dimensional composite image features (non-linear in nature) such that they can be mapped to an existing distance-based index structure without any performance penalty.

The rest of the paper is organized as follows. In Sect. 2 we review the related work in the areas of dimensionality reduction, image similarity measurement, and distance-based access methods. In Sect. 3, we briefly review feature extraction techniques and follow on with detailed presentation of our proposed method. Implementation and experimental results are given and discussed in Sect. 4. Finally, in Sect. 5, we present the conclusions and outline future research.

2 Background

2.1 Image feature dimension reduction

In any imaging system, image features that are extracted by different image processing techniques are often high-dimensional because of the large number of parameters required to model the features. Some parameters in these models are redundant for content-based retrieval purposes, but detecting such redundancies at the image processing stage is not a trivial procedure. Since low-dimensional representations of feature vectors are more efficient for image retrieval from an image database, it is necessary to apply a dimensions reduction technique to eliminate the redundancies (correlated information) of image features as a post-process of feature detection. The goal of a feature dimensions reducer is to discover complex dependencies among the features of images, eliminate correlated information or noise while maintaining sufficient information for discrimination between images of different classes.

Many dimensions reduction methods have been proposed which can be broadly classified into two categories: linear dimensions reduction (LDR) and non-linear dimensions reduction (NLDR).

LDR is well known as an effective process for mapping the original high-dimensional features into low-dimensional ones by eliminating the redundant information from the original feature space. The most well-known statistical approach for doing this is the principal component analysis (PCA) [14,17]. The advantage of the PCA transformation is that it is linear and that any linear correlations present in the data are automatically detected. If the data are known to come from a well-defined model where the underlying factors satisfy various assumptions, then factor analysis can be used to approximate the original data in terms of the common factors and thus can be used to achieve a reduction in dimensions [21]. Multidimensional scaling (MDS) is another well-known LDR technique for discovering the underlying spatial structure of a set of data items from the similarity information among them [18].

Because of the simplicity in the underlying idea of LDR, it is commonly chosen for feature dimensions reduction. For example, the QBIC system [22] used the PCA to reduce a 20-D moment-based shape feature vector for indexing in its image database; Faloutsos and Lin [10] used MDS for indexing and visualisation of a multimedia database.

LDR works well for data that exhibit some linear correlation, for then a small number of eigenvalues may account for a large proportion of the variance of the data, and so dimensions reduction can be achieved. If the data exhibit some non-linear correlation then this is not picked up by LDR. Since image visual features are non-linear in nature, a much better performance in dimensions reduction is expected by using NLDR. The basis of NLDR is the standard non-linear regression analysis as used in neural network approaches, which have been widely studied in recent years. The advantage of using neural network for NLDR is that it can learn directly from the training samples to form a model of the feature data (i.e., the features that matter the most in forming the expected solutions). Since neural network is the core technique that we adopted for doing NLDR in our research work, we will cover this topic in more detail in Sect. 3.

In general, the main difference between LDR and NLDR is that NLDR enables the system to maintain a great deal of knowledge about the information on the data source. This information can be represented as network weights between units in successive layers of the network. Thus, NLDR can be used for reducing the dimensions of image feature vectors that cannot be handled by LDR. The only drawback of NLDR is that the network training process can be very slow.

2.2 Image similarity measurement

A major goal of content-based retrieval is finding the best matched (most similar) images from the multimedia database with respect to a query object (image). The query object can be specified by a sample object (image), descriptive concepts (keywords), or numerical specification. The feature vectors (mainly numerical) for the given query object is then derived using basic image processing techniques such as segmentation and feature extraction. Calculating the similarity between a query object and an object in the multimedia databases is then reduced to computing the distance between two image feature vectors. Given two n-D image feature vectors $\mathbf{x} = (x_1, x_2, \cdots, x_n)^\top$ and $\mathbf{y} = (y_1, y_2, \cdots, y_n)^\top$, where $^\top$ denotes vector and matrix transpose, a similarity function $S(\mathbf{x}, \mathbf{y})$ can be defined using one of the following well-known distance functions:

1. City-block (the L_1-norm): $S(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} |x_i - y_i|$
2. Euclidean (the L_2-norm): $S(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$
3. Minkowski (the L_p-norm): $S(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{N} |x_i - y_i|^p\right)^{\frac{1}{p}}$
4. Dominance (the L_∞-norm): $S(\mathbf{x}, \mathbf{y}) = \max |x_i - y_i|$

Each of the distance functions above has its advantages and disadvantages when applied to image retrieval. For example, the L_1-norm may cause false dismissals (i.e., not all qualified images are retrieved); the L_2-norm, on the other hand, may have false alarms (i.e., unqualified images can also be returned) [28].

So far, research has been focused on finding a similarity function that corresponds only to single features (features of one type, e.g., features that are composed of colour information only or texture information only). That is, only simple queries, such as how similar two images are in terms of colour, are well supported. In [5], the similarity measure of a pair of images based on composite feature vectors described by both texture and colour was proposed as a linear combination of the similarity measure of the individual single feature vector. Their proposal can be detailed as follows: let $\{\mathbf{x}_c, \mathbf{x}_t\}$ and $\{\mathbf{y}_c, \mathbf{y}_t\}$ be the colour and texture feature vectors that fully describe two images X and Y, then the similarity measure of images X and Y, denoted as $\tilde{S}(X, Y)$, is given by:

$$\tilde{S}(X, Y) = \sqrt{\alpha S_c(\mathbf{x}_c, \mathbf{y}_c) + \beta S_t(\mathbf{x}_t, \mathbf{y}_t)} \qquad (1)$$

where the S_c and S_t are the colour and texture similarity functions, respectively; and α and β are non-negative weighting factors. However, criteria for selecting these weighting factors are not mentioned in their research work. From the statistics viewpoint, by treating the above weighting factors as normalization factors, the above definition is just a natural extension of the Euclidean distance function to a high-dimensional space in which the coordinate axes are not commensurable. If the kth weighting factor is set to the inverse of the variance of the kth component of the feature vectors then the distance function is called the *Karl Pearson distance*; if the kth weighting factor is set to the inverse of the range of values for the kth component of the feature vectors then the distance function is said to be *standardized by range*; if correlation was found to be present among the components of the feature vectors then the Mahalanobis distance function can be used [21].

The question that remains to be answered is whether a Euclidean distance function for the similarity measure best correlates with the response from human visual perception in classifying images. That is, when humans perceive two images as similar in colour and in texture, can a distance function given in the form of (1) be defined? Does this same function hold for another pair of images that are also perceived as similar in colour and in texture? So far, no experiments have been conducted that demonstrate (or counter-demonstrate) whether linear combinations of different image features are valid similarity measure based on human visual perception. The importance of designing a distance function that best mimics human perception to approximate a perceptual ordering of the database is not unrecognized. Jain [25] reported that an image database should use human pre-attentive similarity as much as possible; also, the distance functions of QBIC [13] were intended to reflect human perception. Incorporating human visual perception into image similarity measurement is the other major motivation behind our work. This will be discussed in Sect. 3.

2.3 Distance-based access methods

Several spatial access methods have been proposed recently. These methods can be broadly classified into the following classes: *point access methods* and *rectangle access methods*. The point quad-tree, which was first proposed in [12], is an example of a point access method. To handle complex objects, such as circles, polygons, and any undefined irregularly-shaped objects, minimum bounding rectangles (MBRs) have been used to approximate the representations of these objects. Hence, the name, rectangle access method. The K-D-B tree [23], and R^+-tree [26] are some typical examples. A comprehensive survey on SAMs can be found in [24].

The applicability of SAMs is limited on two counts. First, objects for indexing must be represented by feature values in a multi-dimensional space. Second, the design of SAMs is based on the assumption that the comparison of feature values has a negligible CPU cost with respect to disk I/O cost. Unfortunately, in multimedia applications, the assumption above does not normally hold. Consequently, a more general approach to the "similarity indexing" problem has gained popularity in recent years, leading to the development of the so-called *metric trees*. Metric trees only consider the relative distances of objects (rather than their absolute positions in a multi-dimensional space) to organize and partition the search space. The only requirement is that the distance function must be metric so that the triangle inequality property applies and can be used to prune the search space. Several metric trees have been developed so far, including the vp-tree [4], the $GNAT$ [2], the mvp-tree [1], and M-tree [7].

Our goal is not to develop a new indexing structure for high-dimensional image features but to use an existing one effectively. We chose a very well-established access method called the M-tree as the underlying method for indexing our reduced composite image visual features. The M-trees are balanced, paged metric trees which are implemented based on the $GiST$ (*Generalized Search Tree*) [16] framework. Since the design of the M-trees is inspired by the principles of metric trees and database access methods, performance optimization concerns both CPU (distance computations) and I/O costs. In an M-tree, the leaf nodes store all indexed (database) objects represented by their keys or features; the internal nodes store the so-called *routing objects*. A routing object is a database object to which a routing role is assigned by a specific promotion algorithm. See [7] for more details about the design and implementation of M-trees.

3 Hybrid dimension reducer

Multimedia visual features are usually complex and cannot be represented by single feature vectors. Thus, an effective content-based retrieval system cannot be achieved by considering only a single type of feature such as colour, texture or shape alone. However, creating an index based on a concatenation (see (2)) of feature vectors (such as colour, shape, and texture) will result in a very high dimensional feature space, rendering all existing indexing methods useless.

We need to "fuse" the multiple single feature vectors into a composite feature vector which is *low* in dimensions and yet preserves all the necessary information for image retrieval. In this section, we describe our proposed hybrid method of dimensions reduction on image visual features.

3.1 Composite image features

The image features that we deal with in this paper are colour and texture features. Note that our system is not limited to dealing with these two features only. We restrict ourselves to these two visual features for simplification in setting up the experiments and the availability of the source codes for automatic extraction of these two types of features.

3.1.1 Colour features

It is known that the human eye responds well to colour features. In this work, the colour features were extracted using the colour histogram technique [1][29]. Given a discrete colour space defined by some colour axes, the colour histogram is obtained by discretising the image colours and counting the number of times each discrete colour occurs in the image.

In our experiments, we used the colour space CIE L*u*v. The reason for selecting the CIE L*u*v instead of the normal RGB or other colour spaces is that it is more uniform perceptually. We first divided the three axes of the L*u*v space into four sections to obtain a total of 64 (i.e., $4 \times 4 \times 4$) bins for the colour histogram. However, we found that, for the collection of images used in our experiments, not all the bins had non-zero counts. So, after, eliminating those bins which had a zero count, our colour features are presented as 37-D vectors.

3.1.2 Texture features

Texture features carry the property measures, such as the *smoothness*, *coarseness*, and *regularity*, of an image. In this work, the texture features were extracted using a filter-based method. This method detects the global periodicity of intensity values in an image by identifying regions that have high energy, narrow peaks. The advantage of the filter-based methods is in their consistent interpretation of feature data over both natural and artificial images.

The Gabor filter [30] is a frequently used filter in texture extraction. It measures a set of selected orientations and spatial frequencies. Six frequencies are required to cover the range of frequencies from 0 to 60 cycles/degree for human visual perception. We chose 1, 2, 4, 8, 16, and 32 cycles/degrees. The total number of filters needed for our Gabor filter is 30. Texture features are therefore represented as 30-D vectors.

When forming composite feature vectors from the two types of features described above, the most common approach is to use the direct sum operation. Let $\mathbf{x}_c$ and $\mathbf{x}_t$ be the colour and texture feature vectors, the direct sum operation, denoted by the symbol $\oplus$, of these two feature vectors is defined as follows:

$$\mathbf{x} \equiv \mathbf{x}_c \oplus \mathbf{x}_t = \begin{pmatrix} \mathbf{x}_c \\ \mathbf{x}_t \end{pmatrix} \tag{2}$$

The number of dimensions of the composite feature vector $\mathbf{x}$ is then the sum of those of the single feature vectors, i.e., $\dim(\mathbf{x}) = \dim(\mathbf{x}_c) + \dim(\mathbf{x}_t)$. The $\oplus$ operator given in (2) extends naturally to multiple single feature vectors.

3.2 Architecture of hybrid image feature dimension reducer

With the 67-D feature vectors (37 dimensions for colour and 30 dimensions for texture) in our system, the PCA is useful as an initial dimensions reducer while further dimensions reduction for non-linear correlations can be handled by NLDR. Figure 1 shows the overall architecture of our hybrid method.

[1] Part of the source codes for the colour extraction was supplied by the National University of Singapore.

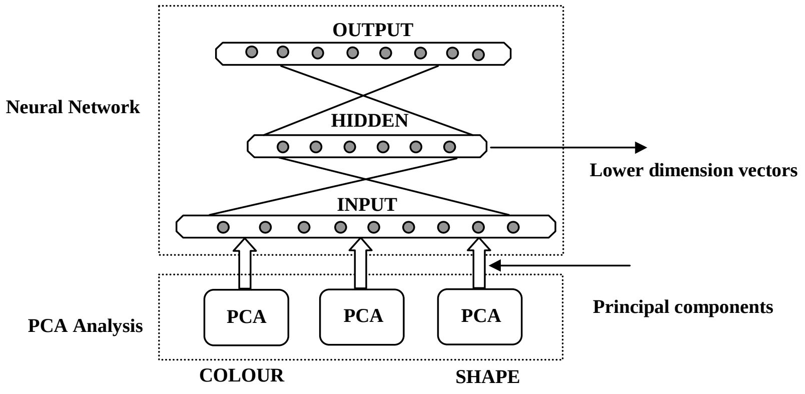

Fig. 1. A hybrid image feature dimensions reduction scheme. The linear PCA appears at the bottom, the non-linear neural network is at the top, and the representation of lower dimensional vectors appears in the hidden layer

The different components of the architecture will be covered in detail in this section.

There are two methods for combining the PCA and NLDR:

1. Apply the PCA to the single feature vectors separately. The lower-dimensional single feature vectors are then combined to form low-dimensional composite feature vectors for NLDR and classification.
2. Apply the PCA to the high-dimensional composite feature vectors. The reduced-dimensional composite feature vectors are then used for NLDR and classification.

Both methods were adopted in our system so that the differences in the reduction results could be compared.

3.2.1 The PCA for dimensions reduction

Mathematically, the PCA method can be described as follows: given a set of N feature vectors $\{ \mathbf{x}_k = (x_{k1}, x_{k2}, \dots x_{kn})^\top \in \mathcal{R}^n \mid k = 1 \cdots N\}$ and the mean vector $\overline{\mathbf{x}}$ computed as $\overline{\mathbf{x}} = \frac{1}{N}\sum_{k=1}^{N} \mathbf{x}_k$. The covariance matrix S is given as

$$S = \frac{1}{N} \sum_{k=1}^{N} (\mathbf{x}_k - \overline{\mathbf{x}})(\mathbf{x}_k - \overline{\mathbf{x}})^\top .$$

Let $\mathbf{v}_i$ and λ_i be a pair of eigenvector and eigenvalue of the covariance matrix S. Then $\mathbf{v}_i$ and λ_i satisfy the following:

$$\lambda_i = \sum_{k=1}^{N} (\, \mathbf{v}_i^\top (\mathbf{x}_k - \overline{\mathbf{x}}) \,)^2 .$$

Since $\mathrm{trace}(S) = \sum_{i=1}^{n} \lambda_i$ accounts for the total variance of the original set of feature vectors, and since λ_i can be arranged in decreasing order, i.e., $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$, if the m (where $m < n$) largest eigenvalues account for a large percentage of variance, then, with an $n \times m$ linear transformation matrix T defined as:

$$T = [\, \mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m \,] , \tag{3}$$

the $m \times n$ transformation $T^\top$ transforms the original n-D feature vectors to m-D ones. That is,

$$T^\top (\mathbf{x}_k - \overline{\mathbf{x}}) = \mathbf{y}_k , \quad k = 1 \cdots N \tag{4}$$

where $\mathbf{y}_k \in \mathcal{R}^m, \forall k$. The matrix T above has orthonormal columns because $\{\mathbf{v}_i \mid i = 1 \cdots n\}$ form an orthonormal basis, i.e.,

$$\mathbf{v}_i^\top \mathbf{v}_j = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{otherwise,} \end{cases}$$

and

$$\|\mathbf{v}_i\| = 1, \quad \forall i.$$

The key idea in dimensions reduction of the PCA is in the computation of λ and the user-determined value m, and finally the $m \times n$ orthogonal matrix $T^\top$, which is the required linear transformation. The feature vectors in the original n-D space can be projected onto an m-D subspace via the transformation $T^\top$. The value of m is normally determined by the percentage of variance that the system can “afford” to lose.

The ith component of the $\mathbf{y}_k$ vector in (4) is called the ith *principal component* (PC) of the original feature vector $\mathbf{x}_k$. Alternatively, one may consider just the ith column of the T matrix defined in (3), then the ith principal component of $\mathbf{x}_k$ is simply

$$y_{ki} = \mathbf{v}_i^\top (\mathbf{x}_k - \overline{\mathbf{x}})$$

where $\mathbf{v}_i$ is the ith eigenvector of S.

The PCA has been employed to reduce the dimensions of single feature vectors so that an efficient index can be constructed for retrieval in the image database [19,8]. It has also been applied to image coding, e.g., for removing correlation from highly correlated data, such as face images [27]. In our work, the PCA is used as the first step in an NLDR method where it provides optimal reduced dimensional feature vectors for the 3-layer neural network, and thus speeds up the NLDR training time.

3.2.2 Classification based on human visual perception

A major part of the human perceptual process involves relating new stimuli to past experiences and trying to answer such question as “Have I ever seen something like this before?” or “What kind of thing is it?”. The Gestalt psychologists maintained that one of the major tasks perceptual processes must perform is the recognition of shapes or form. That is, we tend to perceive whole objects even when we are looking at only a part or some component of that object. Closure, continuity, proximity, and similarity are the four Gestalt principles of perceptual organization that have been applied quite successfully in feature detection and scene understanding in machine vision. Linking and merging a set of detected edge elements into more prominent features such as line and curve segments [3] is a typical application of perceptual organization. Distinguishing figure from ground is another basic and powerful Gestalt principle of visual perceptual organization. When we are presented with an image, we tend to see "things". We interpret the visual message transmitted from the retina to the brain as objects against a background. Even though the image could be as complicated as a ship standing out against the background of sea and sky, a camel and a man standing out against a background of desert sand, or a group of people posing against a background of hills, trees, and a waterfall, our perceptual system does not seem to have any major difficulty in determining which is figure and which is ground [20]. Furthermore, we would distinguish an image of a camel against a background of desert sand as more similar to an image of a camel and a man against the same background than to an image of a camel against a sandy beach. In general, we incorporate all the information about colour, texture, and shape under a certain context that is presented to us and classify the image into the appropriate class.

In conducting our experiments on image classification based on human perception, we first prepared a set of images (there were 163 images altogether), which we called `test-images`, from our 10,000 image collection. This set covers all the 14 different classes of images in the collection. Amongst the images in `test-images`, images in each class have a similarity to each other both in colour and in texture.

We set up a simple image classification experiment on the Web and asked seven people (subjects), all of whom are from different backgrounds, to participate in the experiments. At the beginning of each experiment, a query image was arbitrarily chosen from `test-images` and presented to the subjects. The subjects were then asked to pick 20 images which were most similar in both colour and texture to the query image. Those images that were selected by more than three subjects were classified into the same class as the query image and were then deleted from `test-images`. The experiment was repeated until every image in `test-images` had been categorized into an appropriate class.

The end result of the experiments is that images which are similar to each other in colour and in texture are put into the same class based on human visual perception. These classification results are used in the NLDR process described below.

3.2.3 Neural network for dimension reduction

The advantage of using neural networks for NLDR is that neural networks can be trained from the input data to get to the desired solution. In our work, a three-layer perceptron neural network with a quickprop learning algorithm [9] is used to perform dimensions reductions of image features. In fact, the network acts as an image classifier. In [32], a special neural network called *learning based on experiences and perspectives* (LEP) has been used to create categories of images in the domains of human faces and trademarks; however, no details are given in his work on how the training samples were created. In our system, the training samples were training patterns of the form $(\mathbf{v}, c)$ where $\mathbf{v}$ is a feature vector, which can be either a single-feature vector or a composite feature vector, and c is the class number to which the image represented by $\mathbf{v}$ belongs. We note that the class number for each feature vector was determined by the experiments mentioned in the previous subsection.

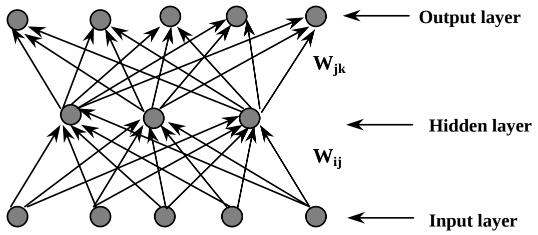

Fig. 2. Layout of a three-layered neural network system

Figure 2 depicts the three-layer neural network that we used. The units in the input layer accept the feature vector $\mathbf{v}$ of each training pattern; the number of units in this layer therefore corresponds to the dimensions of $\mathbf{v}$. The hidden layer is configured to have fewer units. The number of units in the output layer corresponds to the total number of image classes M. Given that $(\mathbf{v}, c)$ is a training pattern, the input layer will accept vector $\mathbf{v}$ while the output layer will contain $(0, \cdots, 0, 1, 0, \cdots, 0)^\top$, which is a vector of dimensions M and has a 1 for the cth component and 0s everywhere else.

Each unit i in the neural network is a simple processing unit that calculates its activation s_i based on its predecessor units p_i, and the overall incoming activation of unit i is given as:

$$\text{net}_i = \sum_{j \in p_i} s_j w_{ij} - \theta_i \tag{5}$$

where j is a predecessor unit of i, the term w_{ij} is the interconnected weights from unit j to unit i, and θ_i is the bias value of the unit i. Passing the value net_i through a non-linear activation function, the activation value s_i of unit i can be obtained. The sigmoid logistic function

$$s_i = \frac{1}{1 + e^{-\text{net}_i}} \tag{6}$$

is used as the activation function.

Supervised learning. Supervised learning is appropriate in our neural network system because we have a well-defined set of training patterns. The learning process governed by the training patterns will adjust the weights in the network so that a desired mapping of input to output activation can be obtained.

Given that we have a set of feature vectors and their appropriate class numbers classified by the subjects, the goal of the supervised learning is to seek the global minimum of cost function E:

$$E = \frac{1}{2} \sum_{\mathbf{p}} \sum_{j} (t_{\mathbf{p}j} - o_{\mathbf{p}j})^2 \tag{7}$$

where $t_{\mathbf{p}j}$ and $o_{\mathbf{p}j}$ are, respectively, the target output and the actual output for feature vector $\mathbf{p}$ at node j.

The rule for updating the weights of the network can be defined as follows:

$$\Delta w_{ij}(t) = \eta \, d(t) \tag{8}$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \tag{9}$$

where η is the parameter that controls the learning rate, and $d(t)$ is the direction along which the weights need to be adjusted in order to minimize the cost function E. There are many learning algorithms for performing weight updates. The quickprop [9] algorithm is one of most frequently used adaptive learning paradigms. The weight update can be obtained by the following equation:

$$\Delta w_{ij}(t) = \frac{\frac{\partial E}{\partial w_{ij}}(t)}{\frac{\partial E}{\partial w_{ij}}(t-1) - \frac{\partial E}{\partial w_{ij}}(t)} \Delta w_{ij}(t-1). \qquad (10)$$

Network training and dimensions reduction. The training procedure of the network consists of repeated presentations of input (the feature vector $\mathbf{v}$'s in the training patterns) and the desired output (the class number c for $\mathbf{v}$) to the network.

In general, the weights of the network are randomly set to small continuous values, initially. Our network adopts the *learning by epoch* approach. This means that the updates of weights only happen after all the training samples have been presented to the network. In the quickprop learning algorithm, there are two important parameters: the learning rate ϵ for the gradient descent and the maximum step size ν. These two parameters govern the convergence of network learning. In general, the learning rate for gradient descent can vary from 0.1 to 0.9. In our system, the learning rate is kept as a constant value during network training. The step size ν is 1.75. In every iteration of the training, the error generated will be in the direction of the minimum error function. This is due to the fact that the training starts in the direction of the eigenvectors associated with the largest eigenvalue for each feature. Thus, the network has less chance of being trapped in a local minimum.

The total gradient error or the total number of error bits indicates the condition of network convergence. When this value does not change during network training, the network is said to have converged. The total error is the sum of the total output minus the desired output. It can be measured by the total number of error bits since the network also functions as a pattern classifier. In this case, the error bit is determined by the difference of the actual and the desired output. If the difference is within $\pm 40\%$, then the number of the error bits is increased by 1.

It is obvious that this hybrid method for dimensions reduction of image features is computationally more efficient than the standard neural network with the original feature vectors. The efficiency is gained by using a relatively small number of network inputs and the network training iterations are conducted in the direction of the largest eigenvalues for each feature.

During the network training process, the network weights gradually converge and the required mapping from image feature vectors to the corresponding classes is implicitly stored in the network.

After the network has been successfully trained, the weights that connect between the input and hidden layers are entries of a transformation that map the feature vectors $\mathbf{v}$ to smaller dimensional vectors. This transformation can be defined as follows: let w_{ij} be the weight that connects the unit j in the input layer and the unit i in the hidden layer; then an image feature vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\top$ is mapped to the units in the hidden layer as:

$$y_i = f\left(\sum_{j=1}^{n} w_{ij} x_j\right), \quad i = 1 \ldots m \qquad (11)$$

where f is the activation function as defined in (6). Here, $\mathbf{y} = (y_1, y_2, \cdots, y_m)^\top$ is an m-vector, and m is the number of units in the hidden layer. Because the number of hidden units (m) is smaller than the number of input units (n), dimensions reduction is achieved from the neural network training process.

Thus, when a high-dimensional feature vector is passed through the network, its activation values in the hidden units form a lower-dimensional vector. This lower dimensional feature vector keeps the most important information of the original feature vectors (colour and texture).

3.2.4 The hybrid training algorithm

The complete training algorithm for this hybrid dimensions reduction method is given as follows:

Step 1: For each type of feature vector, $\{\mathbf{x}_k \in \mathcal{R}^n \mid k = 1 \ldots N\}$, compute the covariance matrix of all the N images.

Step 2: Apply the eigen-decomposition to each of the computed covariance matrix in Step 1. This process yields a list of eigenvectors and eigenvalues (λ), which are normally sorted in decreasing order.

Step 3: Compute the total variance $s = \sum_i^n \lambda_i$ and select the m largest eigenvalues whose sum just exceeds $s * \psi\%$, where ψ is a predefined cut-off value. This step selects the m largest eigenvalues that account for the $\psi\%$ of the total variance of the feature vectors.

Step 4: Construct matrix T using the m corresponding eigenvectors as given in (3).

Step 5: Obtain the new representation $\mathbf{y}_k$ for each image feature vectors $\mathbf{x}_k$ by applying the PCA transformation given in (4).

Step 6: Select the training samples from the image collection. Group these training samples into different classes as determined by the experiments described in Sect. 3.2.2.

Step 7: Construct the composite feature vectors $\mathbf{z}_k$ from the colour and texture feature vectors using the direct sum operation defined in (2).

Step 8: Prepare the training patterns $(\mathbf{z}_k, c_k)$, for all k where c_k is the class number to which the composite feature vector $\mathbf{z}_k$ belongs.

Step 9: Set all the weights and node offsets of the network to small random values.

Step 10: Present the training patterns $\mathbf{z}_k$ as input and c_k as output to the network. The training patterns can be different on each trial; alternatively, the training patterns can be presented cyclically until the weights in the network stabilize.

Step 11: Use the quickprop learning algorithm to update the weights of the network.

Step 12: Test the convergence of the network. If the condition of convergence of the network is satisfied then stop the training process of the network. Otherwise, go back to Step 10 and repeat the process. If the network does not converge, it needs a new starting point. Thus, it is necessary to go back to Step 9 instead of Step 10.

Steps 1–5 cover the dimensions reduction procedure of the PCA, which was applied to all images in the data rather than only to the training samples. This has an advantage in that the covariance matrix for each type of single feature vector contains the global variance of images in the database. The number of principal components to be used is determined by the cut-off value ψ. There is no formal method to define this cut-off value. In Step 3, the cut-off value ψ is set to 99 so that the minimum variance that is retained after the PCA dimensions reduction is at least 99%.

After the completion of the PCA, the images are classified into classes in Step 6. Because the classification incorporates human visual perception, more valid training patterns have been used in the neural network training process. Steps 7–12 then prepare the necessary input and output values for the network training process.

The network training corresponds to Steps 8–11. In general, the weight of each link (a *link* connects two units in the network) is randomly initialized to a small value. The network adopts the *learning by epoch* approach to learning. In the quickprop learning algorithm, the parameter ν that limits the step-size is set to 1.75, and the learning rate for the gradient descent can vary from 0.1 to 0.9. Each time we apply the quickprop learning algorithm, the weight of each link in the network is updated. After a specified number applications of the quickprop learning algorithm, the convergence of the network is tested in Step 12. At this point, it is decided whether the network has converged or a new starting weight is required for each link of the network. In the latter case, the process involved in Steps 9–12 is repeated. The problem about the convergence of a neural network system is still an open one and is outside the scope of this paper.

4 Experiments and discussions

This section presents three experimental results. The aim of these experiments is to demonstrate that the hybrid dimensions reduction method is superior to using the PCA or using neural networks alone. The first experiment shows the result of using the PCA for the reduction of composite feature vectors in images. The second experiment shows the result of using the neural network for reducing the same set of feature vectors in images. The third experiment shows the result of using the proposed hybrid dimensions reduction method.

4.1 Test image collection

We used a collection of 10,000 images for our research. These images were retrieved from different public domains that can be classified under a number of themes which cover natural scenery, architectural buildings, plants, animals, rocks, flags, etc. All the images were scaled to the same size (128×128 pixels).

A subset of this collection of images was then selected to form the training samples (`test-images`). There were three steps involved in forming the training samples. First, we decided on the number of classes according to the themes of the image collection and selected one image for each class from the collection of 10,000 images. This can be done with the help of a domain expert. Next, we built two M-tree image databases for the collection. The first one used colour as the index and the second used texture as the index. For each image in each class, we retrieved the most similar images in colour using the M-tree colour index to form a colour collection of images. We then repeated the same procedure to get images similar in texture for each image in each class to form the texture collection. Finally, we obtained our training samples (there were 163 of them) that are similar both in colour and in texture by taking the intersection of images from the colour and texture collections. The training samples (`test-images`) were presented to the subjects for classification (Sect. 3.2.2).

Appendix A (Table 9) shows the fourteen classes of images categorized by subjects from the image collection. These fourteen classes of images were used in the following experiments.

4.2 The benchmark of the experiments

The aim of these experiments is to determine the accuracy and efficiency of the three methods for dimensions reduction. The images are represented by their corresponding feature vectors (67 dimensions: 37 dimensions for colour; 30 dimensions for texture) which can be viewed as points in a multidimensional feature space. Thus, the distance between any two feature points in this feature space measures the similarity of the two corresponding images. After the dimensions reduction of the image features, a new feature space that combines colour and texture is formed. The distance between two feature points in this space represents the visual similarity of their original images in colour and texture. In order to measure the similarity of images and the separation of classes in this feature space, we introduce the measure *class separation degree* C_i, defined as:

$$C_i = \frac{\sum_{j=1}^{N} Q_j}{N(M-N)}, \quad i = 1 \dots m \tag{12}$$

where m is the number of classes, N is the number of relevant images[2] in the class, M is the total number of test images, and Q_j is the number of images whose distances to the jth image in the class are greater than all the distances from the jth image to its relevant images. Obviously, if C_i is 1 (100%), the ith class is clearly separated from other classes and the images in this class are all similar.

The learning time parameter, t, is used to indicate the efficiency of dimensions reduction, that is, the total number of epochs required for training the dimensions reducer. It is noted that the PCA is performed by the singular value decomposition[3] and so we will not compare its efficiency against the other two methods.

[2] An image is said to be *relevant* to a class if it belongs and has been correctly assigned or classified to that class.

[3] Note that because the covariance matrix is symmetric and positive semi-definite, the singular value decomposition of the covariance matrix is equivalent to the eigen-decomposition of it.

Table 1. The eigenvalues and the percentage of total variation

Class No.	1	2	3	4	5	6	7	8	9	10
λ	1035	636	271	152	140	85	73	64	59	43
%	35.6	21.9	9.34	5.2	4.8	2.9	2.5	2.2	2.0	1.5
Class No.	11	12	13	14	15	16	17	18	19	20
λ	42.3	34.4	30.0	24.7	21.1	19.9	17.2	15.7	13.9	13.3
%	1.4	1.2	1.0	0.8	0.7	0.6	0.59	0.54	0.48	0.46
Class No.	21	22	23	24	25	26	27	28	29	30
λ	12.99	9.78	8.18	6.67	5.97	5.75	5.06	4.85	3.69	3.68
%	0.44	0.34	0.28	0.23	0.21	0.19	0.17	0.16	0.13	0.13
Class No.	31	32	33	34	35	36	37	38	39	40–67
λ	3.52	3.45	3.33	3.19	3.05	2.95	2.74	2.38	2.15	<1.85
%	0.12	0.11	0.11	0.10	0.10	0.10	0.10	0.09	0.08	<0.06

Table 2. Class separation values from the PCA experiment

Class No	1	2	3	4	5	6	7	8
C_i %	60.5	94.9	100	97.9	84.3	100	96.9	95.1
Class No	9	10	11	12	13	14	Average	
C_i %	89.4	91.0	94.5	83.5	90.6	84.1	90.2	

4.3 Result of principal component analysis approach to reduction

In this experiment, the PCA was performed on all training images in Table 9. There are two ways to combine the feature vectors. Let $\mathbf{x}_c$ and $\mathbf{x}_t$ be the colour and texture feature vectors, then the combined feature vectors can be defined as: $\mathbf{x}_c \oplus \mathbf{x}_t$ and $\mathbf{x}_t \oplus \mathbf{x}_c$ (see (2)). We performed the PCA on both combined feature vectors. The results show that there was no difference in eigenvalues for the two different ways in combining the feature vectors. Table 1 shows the eigenvalues and the percentage of total variance. The eigenvalues are arranged in descending order, i.e., $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{67}$. The first 16 eigenvalues account for 94.1% of the total variance of the combined feature vectors. Choosing the first 16 eigenvalues and using the 16 PCs (see Sect. 3.2.1) as a new representation for each of the original 67-D combined feature vectors, we effectively reduced our feature dimensions from 67 to 16.

We note that the 14 image classes are not well separated from each other both before and after the PCA transformation. In the former situation, the image classes reside in a 67-D space; in the latter situation, they are in a 16-D space. To measure the separation of image classes, we selected the first six PCs, which accounted for 79.9% of the total variance of the feature vectors, and computed the class separation value C_i (see (12)) for each class, which is listed in Table 2. It can be seen that only class 3 and 6 are well separated from the other classes. The remaining 12 classes are not well separated in the feature space. If any distance function was applied directly to these 12 classes, the distance between any two images in any one class would be larger than the distance between two images of two different image classes.

4.4 Result of neural network approach to dimension reduction

In this experiment, we used a three-layer neural network discussed in Sect. 3.2.3 to reduce the feature dimensions of the images in `test-images` (see Table 9). All feature vectors were 67-D, containing both colour and texture information from the images. As in the PCA experiment, there are also two ways to combine the colour and texture feature vectors: $\mathbf{x}_c \oplus \mathbf{x}_t$ and $\mathbf{x}_t \oplus \mathbf{x}_c$. The *Recognition Rate* was defined as the percentage of test images that the network could recognize. The learning rate was set to 0.9 and the step size was set to 1.75 in the quickprop learning algorithm (Sect. 3.2.3). The initial weights were chosen randomly within the [0, 0.7] range. The number of hidden nodes was set to 6. Table 3 shows the classification results from the network training process.

The learning time was defined as the average number of epochs required until the network converged. The convergence of the network can be measured by the total error or the total number of error bits of the network. Figure 3 shows the learning time of the network for $\mathbf{x}_c \oplus \mathbf{x}_t$ and $\mathbf{x}_t \oplus \mathbf{x}_c$.

From Fig. 3, we can see that when the network learning is about 6100 ($\mathbf{x}_c \oplus \mathbf{x}_t$) and 5700 ($\mathbf{x}_t \oplus \mathbf{x}_c$) epochs, the errors of the network tend to be steady at about 0.02. Note that it is not necessary to get to zero since an error of 0.02 is already very small in comparison with the initial error. Thus, the network learning times t for $\mathbf{x}_c \oplus \mathbf{x}_t$ and $\mathbf{x}_t \oplus \mathbf{x}_c$ are 6100 and 5700 epochs, respectively.

After the network training was completed, dimensions reduction was achieved by feeding the image feature vectors into the network and taking the vectors computed in the hidden units as the lower dimensional representations. Table 4 shows all class separation values (C_i) measured by the new lower-dimensional representations obtained from this neural network.

In Table 4, it can be seen that all classes of the test image collection are well separated in the new 6-D feature space: the distance of any two images from the same class is less than the distance of any two images from two different classes. However, as shown in Fig. 3, the learning time is very long. In the next section, we show that our proposed hybrid method can improve the network learning time without losing much accuracy.

Table 3. Classification results from the network training process

Class No.	1	2	3	4	5	6	7	8
Recognition Rate($\mathbf{x}_c \oplus \mathbf{x}_t$) %	100	75	100	100	100	100	87	75
Recognition Rate($\mathbf{x}_t \oplus \mathbf{x}_c$) %	100	87	100	100	100	100	100	87

Class No.	9	10	11	12	13	14	Average
Recognition Rate($\mathbf{x}_c \oplus \mathbf{x}_t$) %	87	87	100	100	100	100	93
Recognition Rate($\mathbf{x}_t \oplus \mathbf{x}_c$) %	87	87	100	100	100	100	96

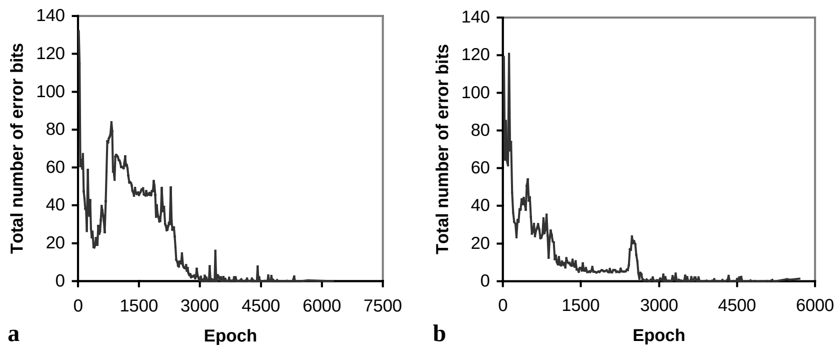

Fig. 3. Learning time of the neural network approach with six hidden units. **a** $\mathbf{x}_c \oplus \mathbf{x}_t$, **b** $\mathbf{x}_t \oplus \mathbf{x}_c$

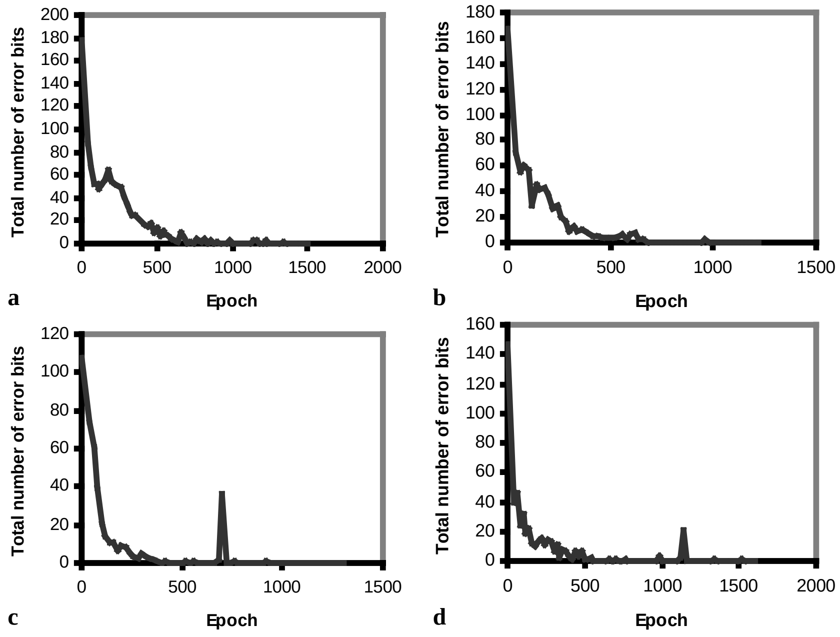

Fig. 4. Learning time of the hybrid approach with six hidden units. **a** $P(\mathbf{x}_c) \oplus P(\mathbf{x}_t)$, **b** $P(\mathbf{x}_t) \oplus P(\mathbf{x}_c)$, **c** $P(\mathbf{x}_c \oplus \mathbf{x}_t)$, **d** $P(\mathbf{x}_t \oplus \mathbf{x}_c)$

Table 4. Class separation values from the neural network experiment

Class	1	2	3	4	5	6	7	8
$C_i(\mathbf{x}_c \oplus \mathbf{x}_t)$ %	100	100	100	100	100	100	100	100
$C_i(\mathbf{x}_t \oplus \mathbf{x}_c)$ %	100	100	100	100	100	100	100	100

Class	9	10	11	12	13	14	Average
$C_i(\mathbf{x}_c \oplus \mathbf{x}_t)$ %	100	99.88	100	100	100	100	99.99
$C_i(\mathbf{x}_t \oplus \mathbf{x}_c)$ %	100	100	100	100	100	100	100

4.5 Result of hybrid approach to reduction

In this experiment, we applied the hybrid dimensions reduction method to the images in the test collection. A dimensions reduction process was first accomplished by applying the PCA to the features of the network training samples. There are four possible ways to obtain the reduced feature vectors: $P(\mathbf{x}_c) \oplus P(\mathbf{x}_t)$, $P(\mathbf{x}_t) \oplus P(\mathbf{x}_c)$, $P(\mathbf{x}_c \oplus \mathbf{x}_t)$ and $P(\mathbf{x}_t \oplus \mathbf{x}_c)$ (see Sect. 3.2) where P denotes the PCA processing. The first 36 PCs, which accounted for about 99.2% of the total variance of the feature vectors in the training samples, were then selected. Thus, the input feature vectors of the network were reduced from 67 to 36 dimensions. Table 5 shows the results of recognition rate from the hybrid network training and Fig. 4 shows the time of the hybrid network learning with six hidden units.

When the network learning time reached 1400 epochs ($P(\mathbf{x}_c) \oplus P(\mathbf{x}_t)$), 980 epochs ($P(\mathbf{x}_t) \oplus P(\mathbf{x}_c)$), 920 epochs

Table 5. Results of recognition rate from the hybrid approach

Class No.	1	2	3	4	5	6	7	8
Recognition Rate($P(\mathbf{x}_c) \oplus P(\mathbf{x}_t)$) %	100	50	100	100	100	100	100	75
Recognition Rate($P(\mathbf{x}_t) \oplus P(\mathbf{x}_c)$) %	100	75	100	100	100	100	100	75
Recognition Rate($P(\mathbf{x}_c \oplus \mathbf{x}_t)$) %	100	75	100	100	100	100	100	75
Recognition Rate($P(\mathbf{x}_t \oplus \mathbf{x}_c)$) %	100	75	100	100	100	100	75	75

Class No.	9	10	11	12	13	14	Average
Recognition Rate($P(\mathbf{x}_c) \oplus P(\mathbf{x}_t)$) %	87	87	100	100	100	100	93
Recognition Rate($P(\mathbf{x}_t) \oplus P(\mathbf{x}_c)$) %	75	75	100	100	100	100	93
Recognition Rate($P(\mathbf{x}_c \oplus \mathbf{x}_t)$) %	75	87	100	100	100	100	94
Recognition Rate($P(\mathbf{x}_t \oplus \mathbf{x}_c)$) %	87	75	100	100	100	100	92

($P(\mathbf{x}_c \oplus \mathbf{x}_t)$), 1600 epochs ($P(\mathbf{x}_t \oplus \mathbf{x}_c)$), the errors of the networks were steady at about 0.02. This indicates that the learning of the networks were completed after 1400, 980, 920, and 1600 epochs for the four methods, respectively. We can see that the learning times are much shorter than the standard network training with input feature vectors being 67 in dimensions. Table 6 shows all the class separation values from this experiment.

From Table 6, we can see that all classes are well separated in the new 6-D feature space, just as in the pure neural network approach, but the learning time is much shorter. There is no difference in the results of the four methods used to organize the input feature vectors.

4.6 Evaluation of reduced dimensional image features using M-trees

We used M-trees [6] for evaluating the quality of our reduced features as indexes. The number of dimensions of M-trees was set to six [4], corresponding to the number of hidden units used in the neural networks. We built three M-tree image databases for the 10,000 image collection using 6-D composite vectors (including colour and texture information after dimensions reduction) of each image in the image collection.

Every image from the collection can serve as a query image. We posed a query image to the M-trees to conduct a k-NN search. Here k was set to 15. The concepts of $Precision$ and $Recall$ in information retrieval were used to evaluate the effectiveness of similarity retrieval. Let P be the number of all images that are relevant to the query image, Q be the number of relevant images retrieved, and R be the total number of images retrieved, then

$$\text{Recall = R} = \frac{Q}{P} \times 100, \quad \text{Precision = P} = \frac{Q}{R} \times 100.$$

A high $Precision$ value means that there are few false alarms (i.e., the percentage of irrelevant images in the retrieval) while a high $Recall$ value means that there are few false dismissals (i.e., the percentage of relevant images which failed to be retrieved). Table 7 shows the results of queries posed against all class images using the three M-trees.

The result in Table 7 shows that for the PCA method, only class 3 and class 6 have no false dismissal. This is the same as the result in Table 2. We can also see that the $Recall$ and $Precision$ values from the neural network and the hybrid methods are almost the same. Thus, the major difference between two approaches is the time required to train the network. One can therefore conclude that it is more advantageous to use a hybrid dimensions reduction method to reduce the dimensions of image features for effective indexing using M-trees.

Figure 6 shows some sample retrieval results from the three M-tree image databases using the same query image (the first one in each result). It is easy to see that using the PCA as dimensions reducer gives the worst result as compared to either neural network or hybrid approach.

We also present a content-based retrieval demonstration system on the web using these three methods. The web site is: http://www.cse.unsw.edu.au/∼imagedb/MVindex/index.html.

4.7 Analysis and discussion

The above experimental results show that the proposed hybrid dimensions reduction method is superior to the other two dimensions reduction methods – the PCA and the neural network – that are applied alone. In this section, we present a discussion of the issues related to the performance of this hybrid method.

4.7.1 Parameters for network training

A wide variety of parameter values were tested in order to find an optimal choice for the network learning algorithm in the above experiments. However, in practice, it is often undesirable or even impossible to perform a large parameter test series. Moreover, different practical applications may require different sets of parameters of the network. In our case, the optimal parameter for the quickprop algorithm is a step size of 1.75 and a learning rate of 0.9.

The number of the hidden units used can also significantly affect the network convergence and learning time. The more the number of hidden units, the easier it is for the network to learn. This is because more hidden units can keep more information. However, since the network is a dimensions reducer, the number of hidden units is restricted to a practical limit. We take $P(\mathbf{x}_c \oplus \mathbf{x}_t)$ in Sect. 4.5 as an example. If we set the hidden units to 15 instead of 6, then the learning time can be reduced dramatically and the network can even reach an error of zero. Figure 5 shows the learning time. It takes only 40 epochs to

[4] M-trees can index up to at least 20 dimensions

Table 6. Class separation values from the hybrid approach

Class No.	1	2	3	4	5	6	7	8
$C_i(P(\mathbf{x}_c) \oplus P(\mathbf{x}_t))$ %	100	100	100	100	100	100	100	100
$C_i(P(\mathbf{x}_t) \oplus P(\mathbf{x}_c))$ %	100	100	100	100	100	100	100	100
$C_i(P(\mathbf{x}_c \oplus \mathbf{x}_t))$ %	100	100	100	100	100	100	100	100
$C_i(P(\mathbf{x}_t \oplus \mathbf{x}_c))$ %	99.9	100	100	100	100	100	100	100

Class No.	9	10	11	12	13	14	Average
$C_i(P(\mathbf{x}_c) \oplus P(\mathbf{x}_t))$ %	100	100	100	100	100	100	100
$C_i(P(\mathbf{x}_t) \oplus P(\mathbf{x}_c))$ %	100	100	99.9	100	100	99.2	99.9
$C_i(P(\mathbf{x}_c \oplus \mathbf{x}_t))$ %	100	100	100	99.9	99.9	100	99.9
$C_i(P(\mathbf{x}_t \oplus \mathbf{x}_c))$ %	100	100	99.8	100	100	100	99.9

Table 7. Results of retrievals using the M-trees

Image class	PCA		Neural Network				Hybrid Method							
			$\mathbf{x}_c \oplus \mathbf{x}_t$		$\mathbf{x}_t \oplus \mathbf{x}_c$		$P(\mathbf{x}_c) \oplus P(\mathbf{x}_t)$		$P(\mathbf{x}_t) \oplus P(\mathbf{x}_c)$		$P(\mathbf{x}_c \oplus \mathbf{x}_t)$		$P(\mathbf{x}_t \oplus \mathbf{x}_c)$	
	R	P	R	P	R	P	R	P	R	P	R	P	R	P
1	32	25	100	80	100	80	100	80	100	80	100	80	100	80
2	85	68	100	80	100	80	100	80	100	80	100	80	100	80
3	100	80	100	80	100	80	100	80	100	80	100	80	100	80
4	97	77	100	80	100	80	100	80	100	80	100	80	100	80
5	76	61	100	80	100	80	100	80	100	80	100	80	100	80
6	100	80	100	80	100	80	100	80	100	80	100	80	100	80
7	88	70	100	80	100	80	100	80	100	80	100	80	100	80
8	93	75	100	80	100	80	100	80	100	80	100	80	100	80
9	82	66	100	80	100	80	100	80	100	80	100	80	100	80
10	76	61	100	80	100	80	100	80	100	80	100	80	100	80
11	78	57	100	73	100	73	100	73	100	73	100	73	100	73
12	61	45	100	73	100	73	100	73	100	73	100	73	100	73
13	81	60	100	73	100	73	100	73	100	73	100	73	100	73
14	82	54	100	67	100	67	100	67	100	67	100	67	100	67
Average	81	63	100	78	100	78	100	78	100	78	100	78	100	78

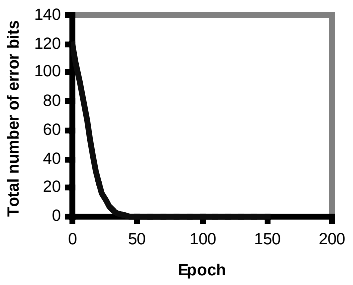

Fig. 5. Learning time of the hybrid dimensions reduction method with 15 hidden units

reach an error of 0.02, compared to Fig. 4 in which about 920 epochs are required.

4.7.2 Number of principal components used in network training

In the hybrid dimensions reduction method, the inputs to the network are not the original image features but the transformed image features from the PCA. The number of PCs selected may affect the network performance. It may not be necessary to take too many PCs for network training. On the other hand, the network may not be trained well with too few PCs since some important information of the feature vectors may have been excluded in the network training process. In this subsection, we give the results of using different numbers of PCs for the hybrid dimensions reduction method for the collection of images in Table 9. Again, we take $P(\mathbf{x}_c \oplus \mathbf{x}_t)$ in Sect. 4.5 as an example. The network training condition is the same as that mentioned in Sect. 4.4 for six hidden units. Table 8 shows the learning time for different numbers of PCs.

It can be seen that the numbers of PCs for the best network training in our application depends on their total variance. There are no significant differences in the time required for network training from 35 to 50 PCs since they account for more than 99% of the total variance. Moreover, since the eigenvalues are in decreasing order, increasing the number of PCs after the first 40 PCs does not require much extra time to train the network. For example, there are only 20 epochs' difference between 45 PCs and 50 PCs. However, if we choose the number of PCs with a total variance that is less than 90% of the total variance then the differences are significant. It takes

Table 8. Learning time of the hybrid approach for different numbers of PCs

Number of PCs	Total variance %	Learning errors	Number of epochs
7	82.4	68.0	>100,000
10	88.2	0.02	11,680
15	93.5	0.02	4,320
20	96.2	0.02	3,040
25	97.7	0.02	1,820
30	98.5	0.02	1,440
35	99.1	0.02	1,180
40	99.5	0.02	780
45	99.7	0.02	820
50	99.9	0.02	840

11,680 epochs for 10 PCs that account for 88.2% of the total variance to reach the ultimate network error of 0.02, which is far greater than the epochs needed for 35 PCs or more.

4.8 Scalability and updates

The number of images that we used in our experiments for testing our dimensions reducer is 10,000, which is a reasonably large image database collection. From our experience, the most time-consuming part of the system is not the neural network training process itself but the collection of training samples for the neural network system. For example, it took us around 25 h to collect a suitable set of training samples (163) from the 10,000 images versus 8 min to train those samples using a Solaris machine with 64 MB RAM. The creation of training samples is a one-off job which can be performed off-line. The indexing structure that we used is the well-known M-tree whose scalability has been demonstrated in many spatial information systems.

The goal of our indexing mechanism is to be able to create a content-based image retrieval system that makes use of human visual perception with a small cost (the initial training). Given an arbitrary query image (i.e., an image not from the database), the system is capable of retrieving images from the database that are most similar in color and texture to this query image. If a new image from the same domain were to be added to the database, then the colour and texture features must be first extracted from the image. The combined colour and texture image features could then be passed through the PCA and neural network for dimensions reduction. Finally, the reduced feature vector could be easily inserted into an M-tree. However, if a new image class from a different domain were to be added, then the neural network system would have to be retrained and the indexes rebuilt for accurate retrieval. Fortunately, for image deletion, the task would be a lot simpler: if an image were to be deleted from the database then all that would be required would be the deletion of the corresponding index from the M-trees.

5 Conclusion

In this paper we have proposed an indexing scheme by combining different types of image features to support queries that involve composite multiple features. The core of this scheme is to combine the PCA and neural network as a hybrid dimensions reducer. The PCA provides the optimal selection of features to reduce the training time of the neural network. Through the learning phase of the network, the context that the human visual system uses for judging the similarity of the visual features in images is acquired. This is implicitly represented as the network weights after the training process. The feature vectors computed at the hidden units (which has a smaller number of dimensions) of the neural network represent our reduced-dimensional composite image features. The distance between any two feature vectors at the hidden layer can be used directly as a measure of similarity between the two corresponding images.

We have developed a learning algorithm to train the hybrid dimensions reducer. We tested this hybrid dimensions reduction method on a collection of 10,000 images. The result is that it achieved the same level of accuracy as the standard neural network approach with a much shorter network training time.

We have also demonstrated the output quality of our hybrid method for indexing the test image collection using M-trees. This shows that our proposed hybrid dimensions reduction of image features can correctly and efficiently reduce the dimensions of image features and accumulate the knowledge of human visual perception in the weights of the network. This enables any existing access method to be used efficiently.

The parameters that affect the network training algorithm is discussed in Sect. 4.7. However, there is a need for further studies on the scalability of the training algorithm. In particular, the issue of how to choose a minimal training set that can be used for a maximal image collection needs to be addressed.

The issues that remain to be studied include extending the experiments to include other visual features such as shape and the topological and spatial relationships of images. There is also a need to investigate more advanced machine learning techniques that can incrementally re-classify images as new images from different domains are added.

A Test-image collection

Table 9 outlines the types of images used in the training and testing process.

B Results of k-NN search using reduced dimensions

Figure 6 shows the results of the k-NN search for the three methods described in the text.

Acknowledgements. We wish to thank the anonymous reviewers for their helpful comments and the editors for their patience while waiting for our revised version. We would also like to thank Ooi Beng Chin from the National University of Singapore for providing the source codes for colour extraction. This research was supported by the Australian Research Council and the Murdoch Special Research Grant MU AMH.D.410 MAR.

Table 9. A collection of 163 images used as a test bed

Image class	Description	No. of training images	No. of testing images
1	Various red flower images similar to each other in colour and in texture.	12	12
2	Various sea scenery images similar to each other in colour and in texture.	12	8
3	Various astronomical images similar to each other in colour and in texture.	12	12
4	Various images of mountains similar to each other in colour and in texture.	12	8
5	Various human face images similar to each other in colour and in texture.	12	8
6	Various images of several bible stories similar to each other in colour and in texture.	12	8
7	Various national flag images similar to each other in colour and in texture.	12	8
8	Various yellow flower images similar to each other in colour and texture.	12	8
9	Various images of artistic works similar to each other in colour and texture.	12	8
10	Various images of green grass similar to each other in colour and texture.	12	8
11	Various animal images similar to each other in colour and texture.	11	11
12	Various sunset scenery images similar to each other in colour and texture.	11	11
13	Various building images similar to each other in colour and texture.	11	11
14	Various images of black-white drawings similar to each other in colour and texture.	10	10

References

1. T. Bozkaya, M. Özsoyoglu (1997) Distance-based indexing for high-dimensional metric spaces. In: SIGMOD'97, pp 357–368, Tucson, Ariz., USA
2. S. Brin (1995) Near neighbour search in large metric spaces. In: VLDB'95, pp 574–584, Zurich, Switzerland
3. J.B. Burns, A.R. Hanson, E.M. Riseman (1984) Extracting straight lines. In: Int. Conf. on Pattern Recognition 1:482–485
4. T. Chiueh (1994) Content-based image indexing. In: VLDB'94, pp 582–593, Santiago, Chile
5. S. Christodoulakis, L. Koveos (1995) Multimedia information systems: issues and approaches. Modern Database Syst., pp 318–337
6. P. Ciaccisa, M. Patella (1998) Bulk loading the M-tree. In: Proc. 9th Australian Database Conf. (ADC'98), Perth, Australia
7. P. Ciaccia, M. Patella, P. Zezula (1997) M-tree: an efficient access method for similarity search in metric spaces. In: Proc. 23rd VLDB Int. Conf., Athens, Greece
8. G.M.P. Euripides, C. Faloutsos (1997) Similarity searching in medical image databases. IEEE Trans. Knowl. Data Eng., 3(9):435–447
9. S.E. Fahlman (1988) An empirical study of learning speed for back-propagation networks. Technical Report CMU-CS 88-162, Carnegie-Mellon University
10. C. Faloutsos, K.I. Lin (1995) FastMap: a fast algorithm for indexing, data mining, and visualization of traditional and multimedia database. In: SIGMOD RECORD, Proc. '95 ACM SIGMOD Int. Conf. on Management of Data, pp 163–174
11. C. Faloutsos, R. Barber, M. Flickner, W. Niblack, D. Peetkovic, W. Equitz (1994) Efficient and effective querying by image content. J. Intell. Inf. Syst., pp 231–262
12. R.A. Finkel, J.L. Bentley (1974) Quad trees: a data structure for retrieval on composite keys. Acta Inf., 4:1–9
13. M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D Petkovic, D. Steele, P. Yanker (1995) Query by image and video content: the QBIC system. IEEE Comput., 28(9):23–32
14. K. Fukunaga, W. Koontz (1970) Representation of random processes using the Karhumen-loève expansion. Inf. Control, 16(1):85–101
15. V.N. Gudivada, V.V. Raghavan (1995) Content-based image retrieval systems. IEEE Comput., 28(9):18–22
16. J.M. Hellerstein, J.F. Naughton, A. Pfeffer (1995) Generalized search trees for database systems. In: 21st VLDB, Zurich, Switzerland, September
17. J. Kittler, P. Young (1973) A new application to feature selection based on the Karhunen-loève expansion. Pattern Recognition, 5
18. J.B. Kruskal, M. Wish (1978) Multidimensional Scaling. SAGE, Beverly Hills, Calif., USA
19. D. Lee, R.W. Barber, W. Niblack, M. Flickner, J. Hafner, D. Petkovic (1993) Indexing for complex queries on a Query-By-Content Image. In: Proc. SPIE Storage Retr. Image Video Database III, pp 24–35
20. R.M. Lerner, P.C. Kendall, D.T. Miller, D.F. Hultsch, R.A. Jensen (1986) Psychology. Macmillan, New York
21. K.V. Mardia, J.T. Kent, J.M. Bibby (1979) Multivariate Analysis. Academic, New York
22. W. Niblack, R. Barber, W. Equitz, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, G. Taubin (1987) The QBIC project: querying image by content using colour,texture and shape. Proc. SPIE, 1908:173–178
23. J.T. Robinson (1981) A search structure for large multimedimensional dynamic indexes. In: Proc. ACM SIGMOD Int. Conf. on Management of Data, pp 10–18
24. H. Samet (1989) The Design and Analysis of Spatial Data Structures. Addison Wesley, Reading, Mass., USA
25. S. Santini, R. Jain (1997) Similarity is a geometer. Multimedia Tools Appl., 5(3):277–306
26. T. Sellis, N. Roussopoulos, C. Faloutsos (1987) The R^+-tree: a dynamic index for multidimensional objects. In: 12th Int. Conf. Very Large Databases(VLDB), pp 507–518
27. L. Sirovich, M. Kirby (1987) A low-dimensional procedure for the identification of human faces. J. Opt. Soc. Am., 4(3):519
28. A.M. Stricker (1994) Bounds for the discrimination power of colour indexing techniques. In: Proc. SPIE Storage Retr. Image Video Database II, pp 15–24
29. M.J. Swain, D.H. Ballard (1991) Colour indexing. Int. J. Comput. Vision, 7(1):11–32
30. M. Turner (1986) Texture discrimination by Gabor functions. Biol. Cybern, 55:71–82
31. D. White, R. Jain (1996) Similarity indexing with the SS-tree. In: Proc. 12th Int. Conf. Data Eng. (ICDE96), New Orleans, USA, pp. 516–523
32. J.-K. Wu (1997) Content-based indexing of multimedia databases. IEEE Trans. Knowl. Data Eng., 9(6):978–989

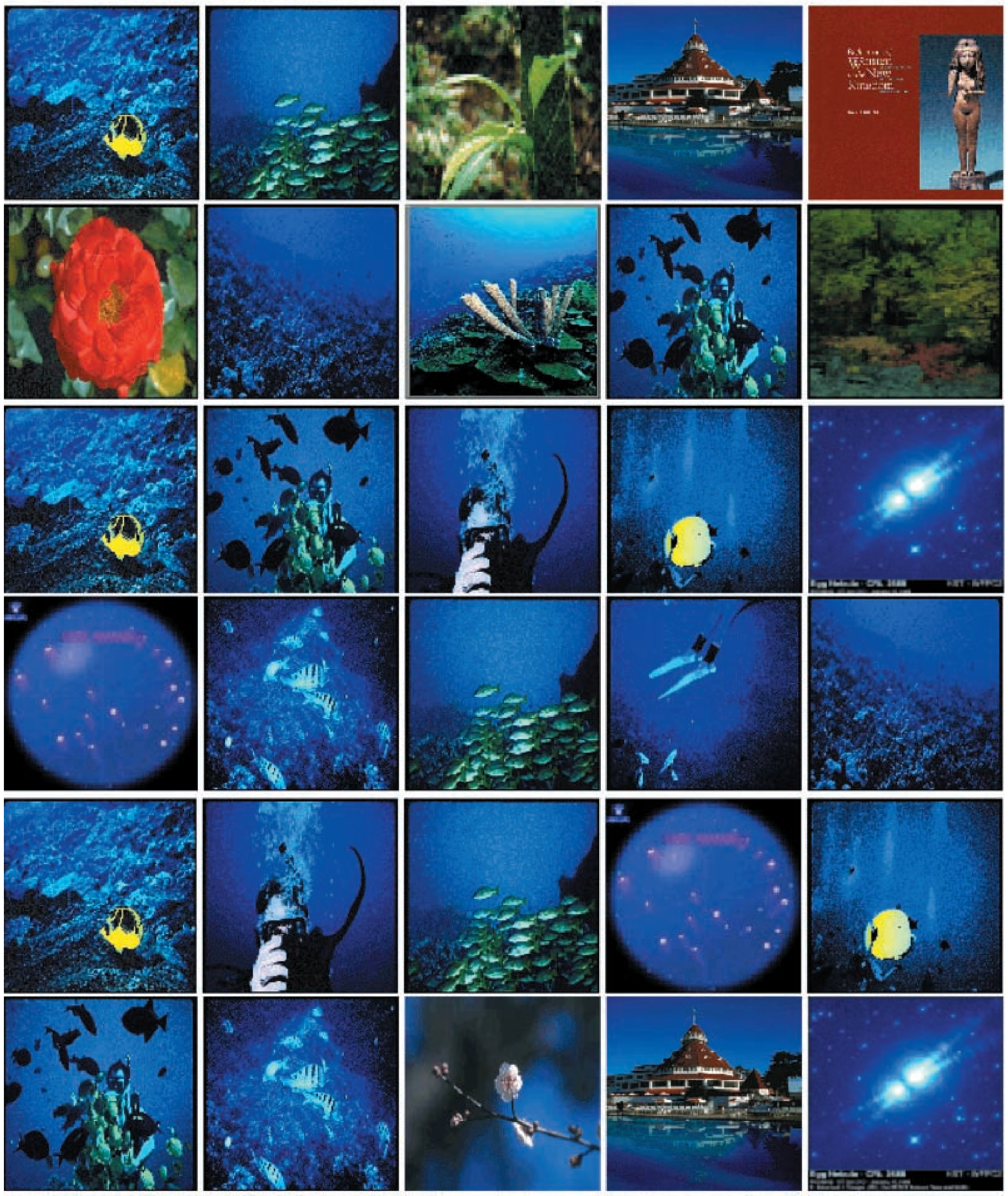

Fig. 6. Results of k-NN search with indexes built using the three methods **a** the PCA, **b** Neural network, **c** Hybrid approach