The University of New South Wales

Reducing Multiple Visual Feature Dimensions for Indexing in Multimedia Databases

by

Rong Lei

A Thesis submitted in fulfillment of the requirements for the
degree of
Master of Engineering
in the
School of Computer Science and Engineering

July 1997

ABSTRACT

Currently there are several popular optimized Spatial Access Methods (SAM) available for multi-dimensional indexing in multimedia databases. However, these approaches depend on two assumptions: a predefined distance function is given and the dimensionality of the image features is low. These assumptions restrict their applications in multimedia database systems due to the fact that a query may use a combination of different sets of features that in turn have very high dimensions. Moreover, it is difficult to obtain a distance function to measure the similarity of human visual perceptions based on the combination of image features.

We propose a fast and robust hybrid method of nonlinear dimension reduction for multiple image features indexing in image databases. This method combines linear statistical and non-linear artificial intelligence techniques to reduce feature dimensions so that the optimized SAM can be applied. Principal Component Analysis (PCA) is the most popular linear dimension reduction method because it is fast and simple. Since the PCA is a linear transformation approach, it can not be applied directly to nonlinear image features. The PCA is used in our method to provide the optimal solution for selecting the best feature to train a nonlinear dimensional reducer. Neural networks are a general method for achieving nonlinear dimension reduction. It can not only reduce the dimension of nonlinear data set but also integrate the knowledge from the training samples to the network with suitable learning methods. In our approach, we use a multi-layers feedforward neural network with supervised learning to perform the nonlinear dimension reduction and the weights are adjusted according to how human perceive the importance of different visual image features.

We demonstrate the results of our method by mapping the reduced dimension feature to existing optimal SAM. It achieves the goal of fast and accurate indexing multiple visual features of images and combines the different features of images into a unique feature space so that the similarity of human visual perceptions can be measured by traditional distance functions.

Acknowledgements

In preparing this thesis, I benefited greatly from a great many people's comments and suggestion, and I would like to express my sincere thanks to them.

I am greatly indebted to my thesis supervisor, Dr. Anne Ngu, and co-supervisor Dr. Jesse Jin for their guidance and support throughout the year of my postgraduate study. Their comments and suggestions have contributed in many ways to the successful completion of this thesis. What I learned from them was both invaluable and memorable which are more than just my research.

I would like to thank my fellow postgraduate candidates and friends for their discussion, suggestion and assistance in this research, in particular Banchong H and Ruth K.

Finally, I would like to express my deepest gratitude to my parent, brother and sister for their endless encouragement and support. This thesis is dedicated to them.

Table of Content

List of tables

List of figures

Chapter 2

Chapter3

Chapter 4

Chapter 5

Chapter 6

Chapter 1

Introduction

1.1 Multimedia Database Systems

The advent of multimedia technology has raised a number of new problems in the field of database management systems. Multimedia technology involves not only conventional data types like characters, numbers and tables, but also media data types such as audio, video, and images. These media data require large storage spaces and their size can vary dynamically. The presentation of these media data types depends on special devices as well as additional information such as image format, compression techniques and layout descriptions. Existing conventional database management systems are not suitable to handle these kinds of data. To manipulate conventional data types combined with media data types, a multimedia database management system is needed. This system should provide the capability of storing, managing and retrieving information based in different media. There are many issues in multimedia database systems such as multimedia information modeling, storing, transportation and retrieval. In this thesis, we are interested in the indexing and retrieval of multimedia data.

The potential applications of multimedia database systems include scientific databases, education, entertainment, weather forecasting, remote sensing, CAD/CAM, art galleries, law enforcement, Internet image archiving, geographic information systems and Internet retailing.

1.2 Motivation

The goal of indexing is to create a compact summary of the database contents to provide an efficient mechanism for the retrieval of information. In a conventional database, index structures are based on the content of the conventional data type, that is, one-dimension. It is also possible to construct an index on the basis of values of two or more fields in combination. The index methods for conventional data types such as B-tree, Hashing and Inverted files have been studied very well and are widely used in commercial DBMS products such as INGRES and ORACLE.

In multimedia database systems, the summary of the image database content consists of the pre-computed feature vectors of images. These image features are multi-dimensional vectors. Indexing will be performed on these summaries of the feature vectors. Traditional indexing structures are unsuitable for indexing multi-dimensional feature vectors. Therefore, to achieve fast and effective retrievals in image database systems, an efficient multi-dimensional indexing scheme is required.

Current image database indexing and querying are based on similarity searches. The idea of indexing the image database is to extract the features from each image in the database and then mapping the features into points in a multi-dimensional feature space. Once this is achieved, the search is based on similarities between the feature vectors by using spatial access methods such as R-tree, linear quadtree and grid files [Faloutsos *et al*., 1994]. However, there are many difficult problems in the area of image feature indexing.

First, image feature vectors have high dimensions. Some of the feature vectors of images can go up to 40-50 dimensions. Existing spatial access methods have an exponential time and space complexity as the number of dimensions increases. For indexing high dimension vectors, the spatial access method such as *R*-tree is no better than sequential scanning. For instance, methods based on *R*-trees can be efficient if the fan-out of the R-tree node remains greater than 2 and the dimensionality is less. The search time with linear quadtrees is proportional to the hypersurface of the query region, while the hypersurface grows geometrically with the high dimensionality. With grid files the directory grows geometrically with the increasing dimensions [Faloutsos *et al*., 1994].

Secondly, the similarity search in images complicates the indexing structure. One of the main differences between an image retrieval system and a traditional database system is the former's ability to rank-order database images by the degree of similarity with the query images [Gudivada & Raghavan, 1995]. Given a set of feature vector Φ, we need to define a similarity function S and a similarity criterion τ. When a query feature vector q is posted onto the image database, a certain number (N) of the feature vectors P_i ($i = 1 \cdots N$) with similarity $S(P_i, q) < \tau$ are returned. If more than one feature set (*i.e.,* $\Phi 1, \Phi 2 \ldots$) is used, we need to define a different similarity function S_i for each feature set. Consequently, for each feature set, a separate index structure is required to support different similarity functions.

Separate index structures for each feature set such as colour, texture or shape can not support queries with composite features efficiently. To answer a query based on composite features, a hierarchical approach is often used where each component of a query is applied against an appropriate index. Then the result is merged in the higher layer and presented to users. 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 colour index, shape index, texture index and then intersecting these three result sets. This is inefficient in terms of storage utilization and system performance. Moreover, this scheme assumes that visual features are equal weight in a complex scene, which might not be true (e.g., the colour of the image might be more prominent than the texture of image in the eye of the beholder).

Although some authors claim to support composite queries by combining different features into an integrated index structure, few of them explain how to integrate different index structures to support composite queries. There are two main problems to be addressed. One is that the integrated feature generates a higher dimensional vector that can not be handled efficiently by existing spatial access methods. The other is the need to measure the similarity in a unique feature space while different visual features are not weighted equally by human visual perceptions.

There are two approaches for solving the first problem. The first approach is to develop a new spatial index method which can handle any dimensional data in any dimensional space. The second approach, which is more practical, is to map the high dimensional feature space into a lower dimensional feature space. Existing spatial access methods can then be used. For the individual feature set, the mapping is straightforward, but for the combined feature set, the mapping is complicated. In this thesis, we aim to investigate the accuracy of indexing images using the second approach.

The second problem is associated with human visual perceptions. The image visual features are not weighted equally in human visual perceptions. For example, the colour, texture and shape of an image have different effects on the human visual system. When those visual features are represented by pre-computed feature vectors, the similarity of images can be measured by using a relative distance between individual image feature vectors. A threshold is used to determine whenever two features are similar.

Euclidean distance is the best-known distance function used to measure the relative distance between image feature vectors. Given two *n*-dimension at feature vectors *V* and *Q*, the distance can be computed as:

$$d(V,Q) = \| V - Q \| = \sqrt{\sum_{i=1}^{n} (v_i - q_i)^2}$$

If the value of *d* is less than threshold τ, then the images corresponding to the feature vectors *V* and *Q* are said to be similar. The assumption behind this is that the feature vectors must be in the same visual feature domain i.e., *V* and *Q* are both either colour vectors, shape vectors or texture vectors. If *V* and *Q* are an integrated combination of different sets of features, even if the *d (V, Q)* is less than the threshold, the corresponding image may not be similar, unless different weight adjustments based on the human visual system are used. To solve this problem, a multi-layer neural network is introduced in this thesis.

The objective of this project is to investigate the index scheme based on the values of two or more visual features in a combination which can efficiently support composite multi-feature queries. The core of the project is to use artificial intelligence techniques to adjust the weighting of the integrated visual features together with feature dimension reduction. The results of our method can be applied directly to existing optimal spatial access methods. It achieves the goal of fast and accurate dimension reduction of multiple visual features of images and combines the different features of images into a unique feature space so that the similarity of human visual perceptions can be measured by traditional distance functions.

1.3 Thesis Organization

In Chapter 2, we review different types of Content-Based Image Retrieval (CBIR) systems. Although many attempts to construct CBIR systems have been made, only recently has attention been focused on integrating the different visual features of images. We briefly look at some existing CBIR system. This thesis does not attempt to cover all aspects of CBIR systems, due to their breadth. It is more concerned with feature selection and indexing. Before proceeding any further, it is necessary to discuss the basic concepts, functions and techniques that are related to CBIR systems. Image feature is first discussed. It includes two visual features, namely colour and texture. Human visual perceptions are also discussed. Then image similarity function is investigated. Spatial Access Methods (SAM) are also reviewed.

Principal Component Analysis (PCA) is a classical statistical technique used for data analysis and dimension reduction, because it is fast and simple. In Chapter 3, we study the characteristics of PCA and discuss the theory for the derivation of PCs. An example of using PCA for image dimension reduction will also be given in this chapter. Since PCA is a linear dimension reduction, the mismatch problem may occur when applying PCA to non-linear image feature vectors. We discuss this specific problem under some conditions and give some suggestions.

A neural network is a general method for achieving nonlinear dimensionality reduction. We use this method to approximate similarity measurements as viewed by the human visual system and to reduce the feature dimension. In Chapter 4, the basic concepts of neural networks are reviewed. Then characteristics of neural networks for image feature dimension reduction are analyzed. A three-layer perceptron neural network used for nonlinear dimension reduction is also discussed. We develop a learning method to train the neural networks based on human visual perceptions. Training time for the neural networks is also addressed since this method can combine the different visual features into a unique lower feature space. The previous two methods have their advantages and disadvantages.

In Chapter 5, we propose a fast and robust dimension-reduction method with weight adjusted for the significant of image features as perceived by humans. This method is based on combining of linear PCA and nonlinear multi-layer neural networks. The training method used for this dimension reduction includes hybrid learning rules. The strength of this method is its efficiency and robustness for multi-features dimension reduction. All associated operations are described in this chapter.

The details of experiments with dimension reduction using the above three different methods are given in Chapter 6. The result of various dimension reduction methods will be shown and discussed. The results of mapping the reduced dimension to R^+-trees are also shown. There is a general discussion of the experiments at the end of this chapter.

Chapter 7 concludes the thesis with a summary of the main contributions of this work and suggestions for future work.

Chapter 2

Background and Related Work

There are many issues that need to be addressed before a practical CBIR system becomes reasonably mature. These issues include similarity-based retrieval, feature extraction, dimension reduction of image features, query specification schemes, data modeling, indexing and so on. Due to the breadth of those issues, we do not attempt to cover all the issues, but concentrate on similarity-based retrieval, feature extraction and indexing.

2.1 Image Features

An image feature is a piece of semantic information extracted from the image. It is very important to extract high quality image features. The quality of features of images should have the following characteristics:

1. Capacity - the number of distinguishable images that can be represented;
2. Simplicity - the feature should be easy to compute;
3. Efficiency - the dimension of feature vector is as low as possible;
4. Maximal Match Number - the maximum number of images a query could possibly retrieve [Sticker, 1994].

Image features can be broadly classified into two categories: low-level features and high-level features. A low-level feature is a statistical attribute of objects in an image such as colour, texture and shape. A high-level feature represents an abstract concept in the image such as a person smiling, a sunset scene, etc. Low-level features can be computed automatically from images. High-level features rely on domain-specific information in the images. Some high-level features can be synthesized from low-level features; others can only be extracted through considerable human involvement. High-level features are normally extracted manually or semi-automatically. It is often a costly operation. Retrieval systems based on high-level features are generally called text-based image retrieval systems and are most suitable for applications involving small numbers of images. Users are domain experts. Content-based image retrieval systems are based on low-level image features.

2.2 Image Similarity Measurement

Images retrieved using a similarity matrix are called content-based retrieval systems. When image features represented by the multi-dimensional vectors are mapped to points in a multi-dimensional feature space, the similarity of images is transformed to the distance of image feature vectors. Given two images associated with their *n*-dimension feature vectors X, and Y, let X = ($\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$) and Y= ($\mathbf{y}_1$, $\mathbf{y}_2$, ..., $\mathbf{y}_n$), $\mathbf{x}_i$,$\mathbf{y}_i$ are *I*-th component of X and Y, respectively, similarity function S (X, Y) has the following criteria:

1. S is a positive function, *i.e.*, S (X,Y) ≥ 0
2. If S (X, Y) = 0, then X = Y
3. Function of S is bi-directional, S (X, Y) = S (Y, X)
4. Function of S meets triangle inequality, S (X, Z) ≤ S (X, Y) + S (Y, Z)

The similarity measures can be broadly categorized as metric-based, set-based, and signal detection based [Jain, 1995]. The metric-based measure is the optimal distance measure for an estimation of image similarity, if signals of the features are corrupted by Gaussian and additive noise. This method is a one-parameter class of the distance function and is widely used for image similarity measurement. It can have several formats such as:

1. L1-norm (city-block): S (X, Y) = $\sum_{i=1}^{N} |x_i - y_i|$
2. L2-norm (Euclidean): S (X, Y) = $\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$

3. Minkowski: $S(X, Y) = (\sum_{i=1}^{N} |x_i - y_i|^r)^{1/r}$

4. Dominance: $S_\infty(X,Y) = \text{Max } |x_i - y_i|$

Every distance function has advantages and disadvantages when applied to image retrieval. *L1*-norm may cause false dismissals (*i.e.,* not all-qualifying images are retrieved) since the *L1*-metric does not take the content of image into account. On the other hand, the *L2*-norm may have false alarms (*i.e.,* non-qualified images can be returned too) [Stricker, 1994].

Variations of the distance function have been used in some projects. In the *QBIC* project of *IBM* [Ashley *et al.,* 1995], the distance function for colour feature vectors is not an Euclidean distance. Instead, they defined the similarity of the function as:

$$S(X,Y) = (X - Y)^T A(X - Y)$$

Where X, Y are histograms of images, T is the transport operator and the location of $a_{i,j}$ in A metric contain the distance between colour *i* and colour *j*. The value of $a_{i,j}$ can be calculated as follow:

$$a_{i,j} = 1 - d\,(c_i - c_j)/d_{max}$$

Where c_i and c_j are the *i*-th and *j*-th colour in colour histograms, and $d\,(c_i - c_j)$ is colour distance, d_{max} is the maximum distance between any two colours. Thus it takes into account the correlation between different components in the colour histogram. The value of *a (i, j)* reflects the degree of similarity between the *i*-th and *j*-th components, the larger value of $a_{i,j}$), the higher the similarity of two components. Although this function can overcome problems of *L1*-norm, it may not be efficient for high dimensional features because the computation of this function is very high. For colour histograms of 256 dimensions, the computation of distance can be expensive.

So far research has focused on finding a similarity function that corresponds only to a single visual feature set. In Leonidas[1994], the similarity distance of a pair of images based on texture and colour is proposed. This distance function is a linear combination of the measure of their corresponding common similarity functions. Let *X*, *Y* be two images. The similarity of two images can be measured as:

$$S(X, Y) = (\alpha S_c(X, Y) + \beta S_t(X,Y))^{1/2}$$

Where the S_c, S_t are colour, texture similarity function respectively; α, β are the weights that are set by some rules. However, these rules are not based on human visual perceptions.

2.3 Feature Extraction

2.3.1 Colour Feature

Colour is a very important visual feature in images. In human perception, colour is an identifiable feature that is local to objects and has the advantage of recognizing an object over shape. The other important property of the colour feature is that the algorithm for colour extraction can be carried out without image segmentation, and it can be performed on either a whole image or objects in the image. There are three colour models namely, RGB, HSI and CMYK (Cyan, Magenta, Yellow and Black). The first two colour models are widely used for image retrieval.

In the RGB model, each colour is specified as a *3*-dimension tuple (R, G, B) which indicates the amount of each primary colour component to be emitted in order to produce the composite colour. Figure 2.3.1 shows the RGB colour space. The advantage of RGB is that it is a unit cube where all possible points inside this cube correspond to realizable colour. However, the disadvantage of this model is that it is not sensible to measure colour differences in RGB since the colour specified in it are not perceptually uniform. The RGB is a commonly used colour space which corresponds to a colour CRT display. In another words, this colour model is device dependent. It means that the same colour may appear different in two different monitors.

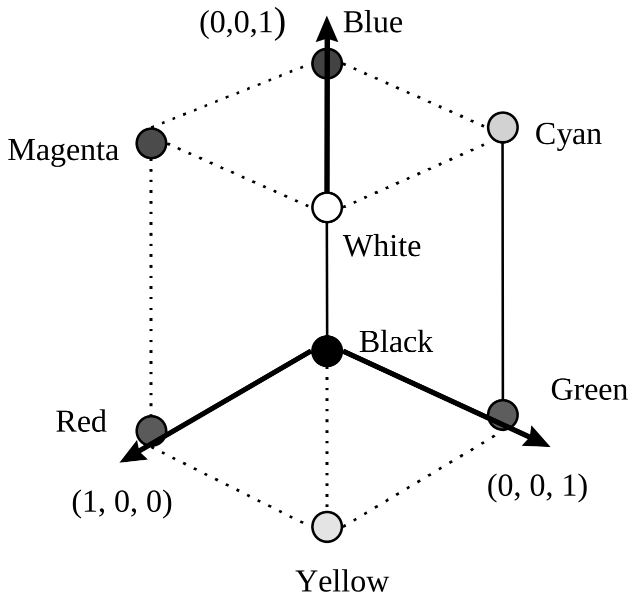


Figure 2.3.1 RGB Colour Space

Colour features of images can also be represented by the HSI colour system. In this system, *H* (hue) represents the colour attribute that describes a pure colour such as pure red, green or gray; *S* (saturation) is a measurement of the degree to which a pure colour is diluted by white light; *I* (intensity) is the gray level light, varying from black to white. Figure 2.3.2 shows the HSI colour space. Since the hue, saturation and intensity are closely related to the attributes of human perception, it is easy for human to tell the three attributes when a colour is perceived, whereas in the case of RGB model, it is difficult for our perceptual power to resolve a colour into its R, G and B components. The other advantage of HSI colour space is that the intensity component is decoupled from the colour information in the image.

The HSI colour systems generate colour features as one-dimension vectors. There are several methods to extract colour features from the image under the HSI system. Some methods can only be used for part of an image and some methods can be used for both whole and partial image.

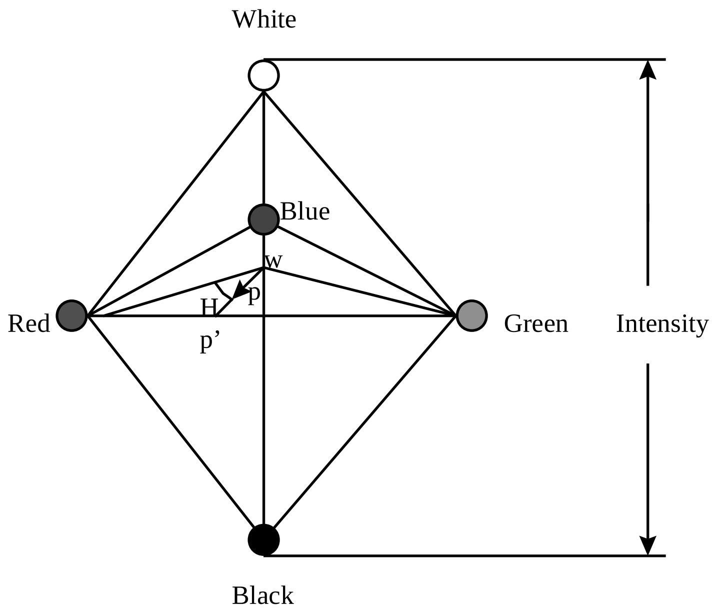


Figure 2.3.2 HSI colour spaces

The Hue of the colour point p in Figure 2.3.1 is the angle of the vector shown with respect to the red axis. The saturation of colour point p is the ratio of |wp| / |wp'| and the intensity of colour point *p* is measured with respect to a line perpendicular to the triangle and passing through its center.

Template matching [Andreadis *et al*, 1990] is one of the most basic methods of extracting the colour features by defining the colour regions in a template image or object and counting the number of pixels that fall into each colour region. The limitation of this method is that the template must be obtained manually and the number of the templates required is quite large.

To overcome the limitation of template matching, Swain [Swain *et al*, 1991] proposed a colour matching method based on histogram intersection. The idea of this method is to find the matches of all colour bins in the colour histograms of the query image and the model. The match value *H* is given by:

$$H(I, J) = \frac{\sum_{k=1}^{n} \min(I_K, J_k)}{\sum_{k=1}^{n} J_K}$$

where n is the total number of bins, and I, J are histograms of the image and the model image in the database respectively. It is shown that if two images are similar, the match value H is close to unity. Since the match value has to be computed for every model histogram during retrieval, this is computationally expensive. Moreover the information on positional distribution of colour in the image may be lost if the colour of the whole image is represented by a single colour histogram [Lu, 1993]. The advantage of the histogram intersection method is that colour histograms are invariant to translation and rotation about the view axis, and can only change slowly under changes of the angle of view, and the change in scale and occlusion. This method was used in the QBIC project [Niblack, 1993].

2.3.2 Texture Feature

Texture is another important visual feature of images. Texture features include characteristics such as smoothness, coarseness and regularity. Texture feature extraction methods can be broadly classified into three categories, namely statistical methods [Gonz 1992, Hara 1979], structural methods [Shao, 1994] and filter-based methods [Gabo 1946, Daug 1985].

The ideas of statistical methods assume that the pixel value of an image is statistically interdependent. The method for generating the texture is totally irrelevant. Thus this method can easily be theoretically analyzed and implemented. For example, moments texture analysis is the simplest approach for texture feature extraction, where the gray-level histogram of an image or region is defined as moments [Gonzalez & Woods 1992]. Different orders of moments give different information about an image. The nth moment of an image is generally computed as follows:

$$\upsilon_n(\aleph) = \sum_{i=1}^{h} (\aleph_i - M)^n \rho(\aleph_i)$$

$$M = \sum_{i=1}^{h} (\aleph_i * \rho(\aleph_i))$$

Where $\aleph_i$ is a random variable denoting discrete image intensity, and $\rho(\aleph_i)$ is the corresponding histogram. The variable h is the number of distinct intensity levels and M is the mean value of $\aleph_i$. However, this is a purely mathematical computation and does not provide enough information on the sense of direction as real moments.

Texture analysis by using the structural approach assumes that every texture can be generated by texture primitives in a regular pattern. But texture primitives may also have texture and need to be extracted. Thus it is impossible to extract the basic element based on this recursive property [Shao & Forstner, 1994]. Therefore this method is not popular in image texture analysis.

Perhaps the most popular method for texture analysis is the filter-based approach. This approach describes image functions by their local amplitude spectra. The global periodicity in an image can be detected by identifying high energy narrow peaks in spectrum. One example of this approach is the two-dimension Gabor filter.

The Gabor filter is one of the frequently used filters in texture extraction. It measures a set of selected orientations and spatial frequencies. It can also minimize the product of effective areas occupied in the 2D space and the 2D-frequency domain. They have been shown to be optimal in the sense of minimizing the join of two-dimensional areas uncertain in space and frequency [Daugman, 1985]. This localization property has been found to be very useful in many image analysis problems. The equation of Gabor can be represented as:

$$g(X,Y) = \exp\{-4\ln 2[(\frac{P_x}{a})^2 + (\frac{P_y}{b})^2] / 2w^2\} * \exp[2\pi fj(x\cos\theta + y\sin\theta)]$$

where θ is the orientation, *f* is the spatial frequency, *w* is the bandwidth, *a* and *b* are the width and height of the filter; and

$$P_x = x\cos\theta + y\sin\theta;\ P_y = -x\cos\theta + y\sin\theta$$

In [Turner, 1986], a set of fixed Gaussian window filters have been used to segment the image texture. The texture edges can also be extracted through a variable set of filters [Shao & Forstner, 1994].

In summary, different assumptions retain to the above three categories of methods for texture analysis. Each of the method has advantages and disadvantages. The use of these methods will depend on a particular application.

2.4 Human Visual Perception

The term perception is commonly used in a number of different ways. To psychologists, perception can be defined as an intermediate step in upward relaying of primitive sensations to higher levels of processing, in accordance with the traditional simple-to-complex hierarchy of sensation-perception-cognition [Bloomer, 1989, *p*.17]. In Biology, perception is the acquisition and processing of sensory information in order to see, hear, taste, smell or feel objects in the world. It guides also an organism's actions with respect to those objects. Perception may involve conscious awareness of objects and events [Skeuler & Blacke, 1994, *p*.515]

Human visual perception is explored as a mediating process which a human's experience of meaning will render unified and coherent [Bloomer, 1989, *p*.17]. It is about how people see objects in the world. Human visual perception presents the richest source of environment information. The visual perceptions mirror the portion of the human brain that is devoted to visual vision.

Human visual perception consists of a sequence, stretching from events in the physical world external to a perceiver. This sequence can be seen in Figure 2.4.1[Skeuler & Blacke, 1994, *p*.2]. There are three processes in human visual perception: detection, discrimination and identification. Detection helps humans pick up distinguish objects from their surroundings. Discrimination can help humans distinguish one object from another and identification can help humans to know exactly what a particular object is. To deal with the environment, people must be able to accomplish all three processes rapidly and accurately [Skeuler & Blacke, 1994, *p.141*]. It should be apparent that these three processes form a hierarchy: discrimination first requires detection, and identification must be preceded by discrimination.

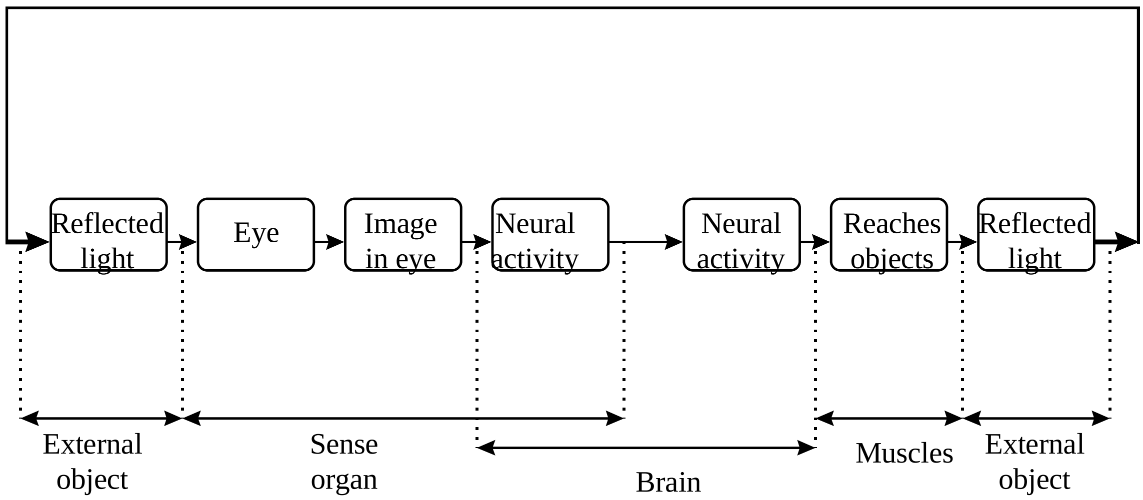


Figure 2.4.1 Sequence of major events involved in human visual perception

An object can differ in many ways from its surroundings. Specifically, the object can differ in colour, shape and texture and different visual features of objects can produce different effects on human visual perceptions. Different people can perceive the same object in different ways. Colour and texture perceptions are two main human visual perceptions.

Colour perception is a very important part of perception. Colour makes it easy for human eyes to pick an object from its background (detection). In addition, colour helps humans distinguish among various objects in the environment (discrimination). The purpose of colour is to help humans detect and discriminate objects. Human colour visual perceptions depend on two factors: 1) the light reflected from objects and 2) the properties of the human eyes and neuron's systems. The light reflected from objects is the wavelength. The human visual system retains information about the wavelength distribution of reflected light as it can usually tell where one object begins and another ends. This wavelength information enables human to recognize what sort of object it has encountered and what action is called for.

It is noted that not everyone sees colour in the same way. People with such variations are described as "colour blind." Colour blindness can be either congenital or acquired during an individual's lifetime. Rarely are people completely colour blind; rather, they do see colour but in a way different from the normal. Sometimes people simply do not notice that their use of names for colour differs from those of other people. The ability to perceive colour help human detect and discriminate the object in their environment [Skeuler & Blacke, 1994, *p.469*]. Because artificial objects or natural objects have characteristic pigments, they absorb and reflect light in characteristic ways.

Texture perception is another important capability of human visual system. The surface of each object has a visual texture, such as the grain in wood. Textures could be discriminated in terms of their mosaic features, which do not necessarily involve figure perceptions yet do incorporate visual feature analysis and synthesis. The discriminable aspects of texture pairs are given by their contract, colour, intensity, and the usual peripheral or optical conditions. The surface texture will appear to vary with human visual perceptions. The human texture perception bridges the gap between specific optical properties of an image (*i.e*., contrast) and the highly qualitative aspects of pattern and figure perception.

The density of a surface's texture appear to vary with distance (called texture gradient). The texture gradient can provide precise and unambiguous information about the distances and slants of surface, as well as about the size of objects located on those surfaces. It can also signal the presence of edges or corners of the object. The texture gradient helps humans perceive the depth of objects which influences perception of their size, resulting in size constancy as well as certain illusions of size.

There are many factors that can affect human perceptions. The familiarity of an object plays an important role in shaping human visual perception experiences. For example, an experienced bird watcher can distinguish among hundreds of different bird species that to most people they look the same. Humans are creatures whose perceptions become differentiated with appropriate training. Simple exposure to a variety of interesting stimuli can sharp one's capacity to discriminate among them. The context of natural scenes and the knowledge of world not only shape perception but also imbue perception with value and utility.

2.5 Indexing in Image Databases

In CBIR systems, queries can generally be classified as equality-match or similarity-match. Equality-match refers to searching for a particular image, which similarity-match refers to searching all the images within a specified range. In general, the basic query classes that will have reasonable generality for dealing with diverse applications in CBIR can be defined as:

- Colour
- Texture
- Sketch
- Shape
- Volume
- Topological relationships
- Browsing
- Objective attributes
- Subjective attributes
- Motion
- Text
- Domain concepts

Colour queries can be constructed by users using a colour picker or by specifying a partial colour histogram of the image [Wu *et al*, 1995, Stricker, 1994]. Texture queries can be specified by using various templates provided by the system interface [Pickard & Minka, 1995, Zhang & Zhong, 1995]. Sketch queries let users outline an object and then retrieve the images containing similar objects from image databases [Ashley, 1995]. A shape query can be constructed using simple painting tools [Mehrotra & Gray, 1995]. Volume queries can be defined as a 3-dimensional shape queries class. Topological relationship queries involve spatial relationships between two or more objects in an image. The relationships can be either directional or adjacency. When users specify the relationships of objects in an image, the system should return images with similar relationships [Lee, 1991, Orenstein & Manola, 1988]. Query by browsing is a major feature for those users who are unfamiliar with the structures and types of information available in image databases. Object, text and domain queries normally appear in text-based retrieval systems such as query by name, date, text, etc. These kinds of queries can be answered by traditional database query languages like Structure Query Language (SQL) [Ogle & Stonebraker, 1995]. On the other hand, the subjective query classes are ambiguous because subjective attributes can be interpreted differently by different users. For example, one might can say the moon is round, while may think it is an ellipse. Motion queries concern retrieval of a series of images such as video segments or camera viewpoints [Flickner *et al.*, 1995].

The above query classes provide a meaningful way for users to retrieve images from databases. A specific query class can be composed with other query classes. For example, users can issue a query such as “retrieve all images containing a red apple on desk.” This includes relationship, colour and texture query classes. It should be noted that this kind of composite query is either transparent to the user or can be constructed by example, *i.e.*, the user submits a query in the form of a prototype image and the system automatically uses the corresponding features vectors for query processing. Query language of conventional databases can not fully support these kinds of images queries. Recently, some image query languages have been proposed. In Joseph and Cardenas [1988], a language called PICQUERY is used to perform several image queries. The major operations include colour transformation, rotation, zooming, shape detection, segmentation, topological relations and similarity retrieval. This language focuses on image manipulations rather then database query operations. An efficient query language of image databases should include not only images manipulation but also traditional database operation.

In an image database system, data items span a multi-dimensional space. Efficient query processing relies on a data structure that can support multi-dimension indexing of these data items. It has been recognized in the past that traditional database systems can support indexing of one-dimensional data items such as integers or real numbers and strings very well. Various data structures have been used to retrieve these one-dimensional data items such as B-trees [Bayer & McCreigh, 1972] [Comer, 1979], ISAM, hashing and binary trees [Knuth, 1973]. However, these techniques could not handle efficiently multi-dimensional data in a multi-dimensional space. For multidimensional data, a method for achieving faster-than-sequential searching is used. This method is called SAM.

Several spatial access methods have been proposed to index multi-dimensional data. We review some of them in this section. A comprehensive survey can be found in [Samet, 1989]. Since the goal of this thesis is not to develop a new indexing structure for multi-dimensional data items but rather to use the existing one efficiently, we will only highlight the strengths and weaknesses of these various access methods.

2.5.1 Point Access Methods

Point is the most common multi-dimensional data that has been studied in the past. The ideas of the point access method can be extended to other multi-dimensional data such as rectangles, circles, polygons etc. The main idea of the point method is that the whole point's space is divided into disjoint subspace so that the subspace contains no more than *n* points, where *n* can be defined by the capacity of a disk page. In general, it is the maximum number of data that a disk page can hold.

2.5.1.1 Point Quad-Tree

The point quad-tree is a multidimensional generalization of the binary search tree. It is first proposed in Finkel and Bentley [1974]. Figure 2.5.1.1.1 shows a two-dimensional quad-tree, in which each node has four sons, namely, NW, NE, SW, and SE.

In general, *k*-dimensional quad-trees may have 2^k sons, which in turn are the roots of subtrees corresponding to the 2^k quadrants. Point quad-trees can support exact match and range queries, but for range query, multiple subnodes of previous visited nodes have to be traversed during searching. If there is a split during insertion, the split is in all dimensions of the tree, and a *k*-dimensional hyperplane (*k-d* cut) is required to perform split operations. The position of the hyperplane for splitting is dynamic and is limited to the region to be split. Since the hyperplane grows geometrically with the dimension, and searching is proportional to the hyperplane of the query region, the searching time also increases geometrically with the dimension. Because data points are stored in the internal nodes, if data points are deleted, related nodes have to be deleted too. Thus the subtrees of this node have to be reorganized *i.e.*, all data points reinserted in the subtree. Therefore, this structure is more suitable for static databases rather than dynamic databases.

2.5.1.2 The *kd*-Trees

Another typical point structure is *kd*-tree, which was first proposed in Bently [1975]. The nodes of *kd*-tree contain data points as well as a discriminator that is used to indicate the key on which searching direction depends. For a *k*-dimensional *kd* tree, the value of the discriminator ranges from *1 - k*. If the discriminator value of the node p is the *i*th attribute of the *k*- dimensional data, then the *i*th attribute of any node of the left-branch of p is less than the discriminator, and the *i*th attribute of any node of the right-branch of p is greater than the discriminator. Thus range queries can be performed more quickly. During insertion, the split is done by one hyperplane that can be pre-determined and limited within the split region.

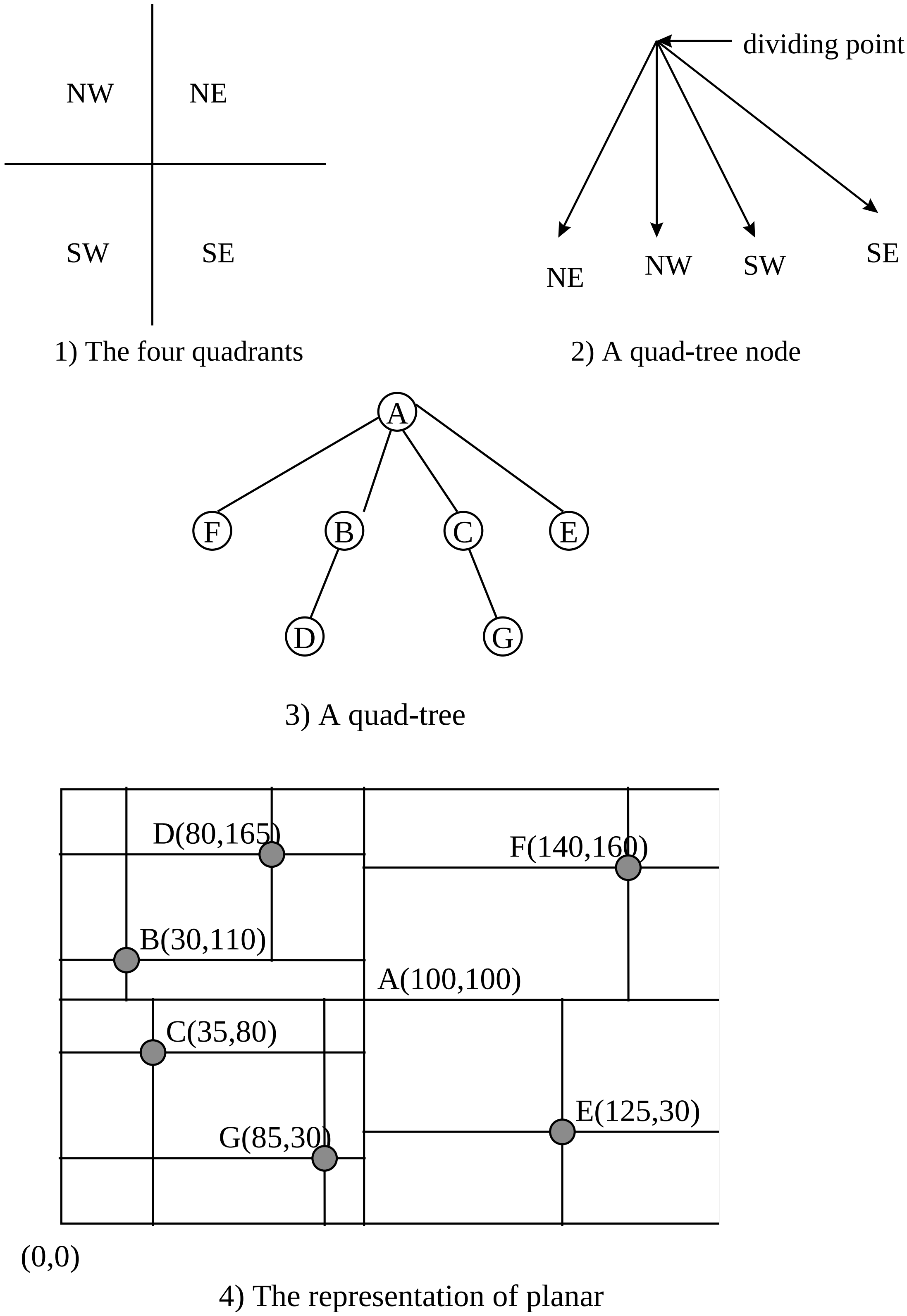
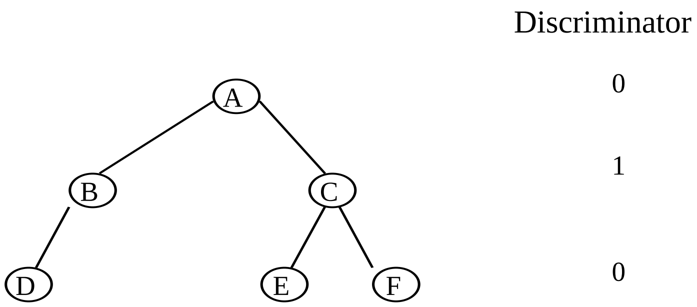
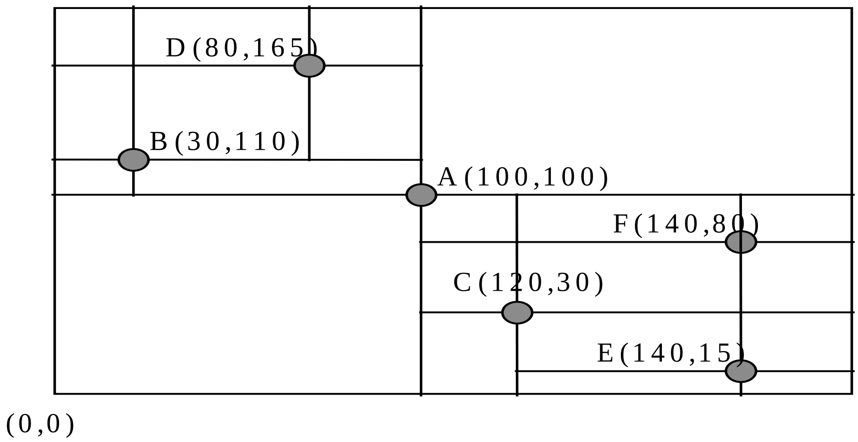


Figure 2.5.1.1.1 A two-dimensional quad-tree.

However, deletion operations of *kd*-trees are difficult since the replacement of the discriminators of the deleted nodes is too costly. The structure of a *kd*-tree is given in Figure 2.5.1.2.1. In recent years, many variants of *kd*-trees such as the *K-D-B*-tree [Robinson, 1981], which is the combination of a *kd*-tree and a *B*-tree are being proposed. The idea of *K-D-B* tree is to store actual data points in the leaf nodes, while the non-leaf nodes hold the descriptions of subspace or the pointers to descendants. The operation of searching and deletion of *kd*-tree is straightforward. However, the insertion into the tree involves not only splitting of leaf nodes but of non-leaf nodes. If a new point is inserted into a full leaf node, the split of the leaf-node starts first and then may propagate non-leaf nodes upward and may go all the way up to the root of tree.



a) The structure of a kd-tree



b) The representation of planar

Figure 2.5.1.2.1 The structure of *kd*-tree

When the split occurs in non-leaf nodes, a one-dimensional pre-defined hyperplane is used to split the space of a region page into two subspaces. The *K-D-B*-tree inherits the "height-balanced" property of the *B*-tree. However, the storage of the *K-D-B*-tree is not as efficient as the *B*-tree and can only be used for zero sized objects. For a non-zero-sized object, a collection of rectangular methods has been proposed.

2.5.2 Multi-Dimensional Access Methods

To handle more complex objects, such as circles, polygons and any undefined irregularly-shaped object, a minimum-bounding rectangle (MBR) can be used to approximately represent those objects. The MBR is the smallest rectangle that encloses the object and it has been widely used in proximity query processing. In general, the methods for handling rectangles can be classified onto three classes, namely, Object Transformation methods, Object Duplication (Object Clipping) methods and Overlapping Bounding Rectangle methods [Ooi, 1988, Seeger, & Kriegel, 1988]. We discuss these methods with some examples.

2.5.2.1 Object Transformation

This method transforms *k*-dimensional rectangles into a higher dimensional (normal *2k*-dimensional) or lower dimensional (*1D*) point. In Orenstein [1984], a method called *Z*-transform has been proposed to transform a *k-D* object into *1-D* line segments by using a space-filling curve. This method tries to optimize the error distance between two points in *k-D* and *1-D* space. For example, a *2-D* rectangle represented as (x_1, y_1, x_2, y_2) can be described as a point in *4-D* space where each direction can be characterized by four coordinates. When this transformation is completed, any point access method can be applied to index these rectangles. Apart from the point access method, Hinrichs and Nievergelt [1983] proposed to use a grid file by rotating the axes. The rotation of axes can avoid non-uniform distribution of points. The structure of the grid file has two components: *n* linear scales and an *n*-dimensional directory (shows in figure 2.5.2.1.1). An *n*-dimensional space is partitioned by an orthogonal grid that is *n* one-dimensional array (scales). Each boundary of the grid forms a (*n - 1*)-dimensional hyperplane that cuts the data space into two subspaces. The boundaries which form *k*-dimensional unpartitioned rectangular subspaces are called the grid directory. The grid cells in this directory contain the address of a data page. The splitting and merging may happen during insertion or deletion if the data page is full or the joint occupancies of two adjacent storage regions drop below a threshold. A major problem of grid file is that the directory grows geometrically with the growth of the dimensions of objects. Thus the retrieval of a non-zero size object can be fairly expensive.

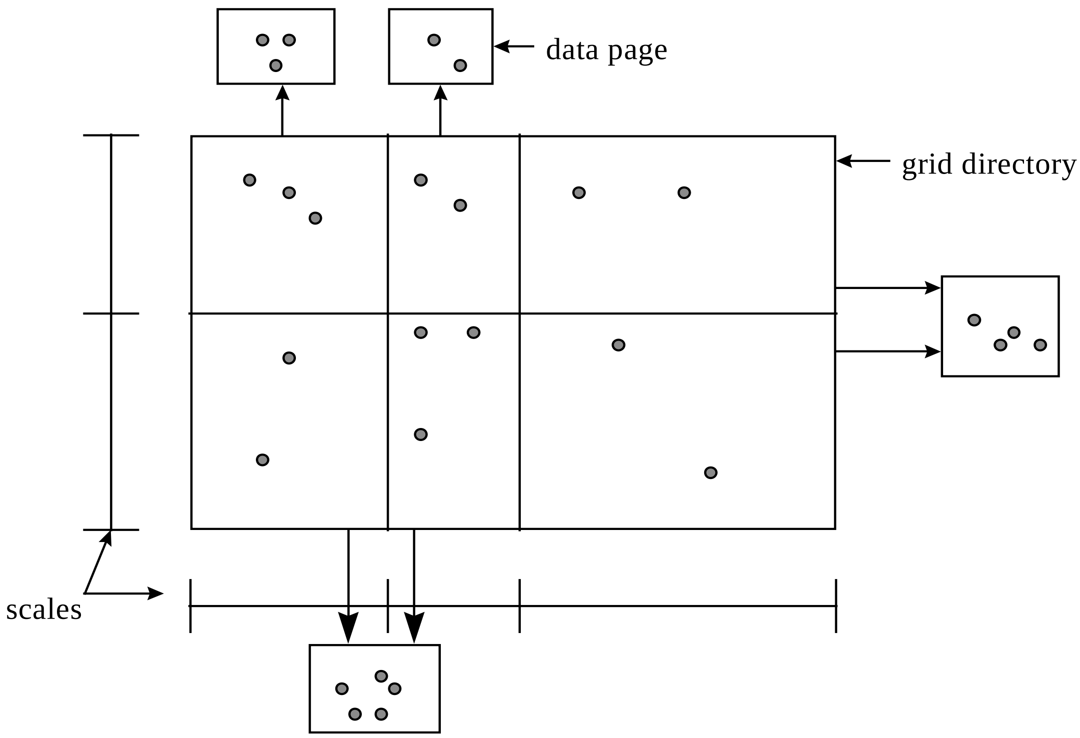
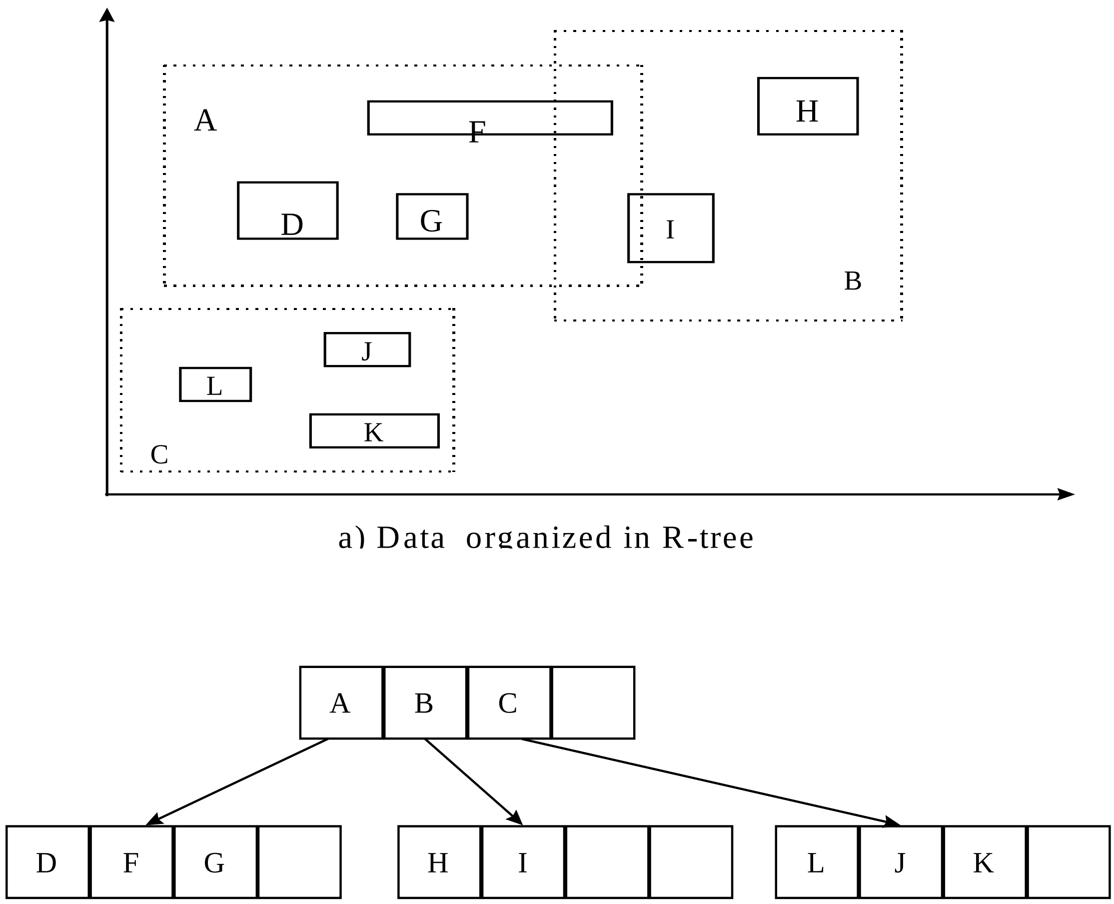


Figure 2.5.2.1.1 The grid file

2.5.2.2 Object Bounding

The idea of this method is to partition an *n-D* space into an overlapping sub-space such that objects are totally included in only one of the subspaces. The first person who proposed the use of overlapping bounding sub-space was Guttman with his *R*-Tree [Guttman, 1984]. The *R*-Tree is a multi-dimensional generalization of the *B*-tree, and hence the tree is height-balanced (all leaf nodes appear on the same level which is a desirable feature) and the space utilization is guaranteed to be at least 50%. In *R*-tree, a leaf node contains entries of the form (object-*id*, *R*) where object-*id* is a pointer to a data object description, and *R* is an *n*-dimensional MBR that bounds its data objects. A non-leaf node contains entries of the form (*ptr*, *R*) where ptr is a pointer to a child node in the *R*-tree and *R* is a MBR that covers all rectangles in the child nodes in the sub-trees. During insertion, the *R*-tree is traversed and all rectangles in the current non-leaf nodes are examined. Figure 2.5.2.2.1 illustrates data rectangles organized in an *R*-tree and the correspondent structure of the *R*-tree.



b) Rtree for the rectangles of a)

Figure 2.5.2.2.1 Data rectangles and structure in an *R*-tree

The deletion operation could cause the leaf nodes to underflow, *i. e.,* the node needs to be deleted and all remaining entries of that node are reinserted from the root. Also, the deletion of an object may change the bounding rectangle of entries in ancestor nodes so that readjustment of these entries is required.

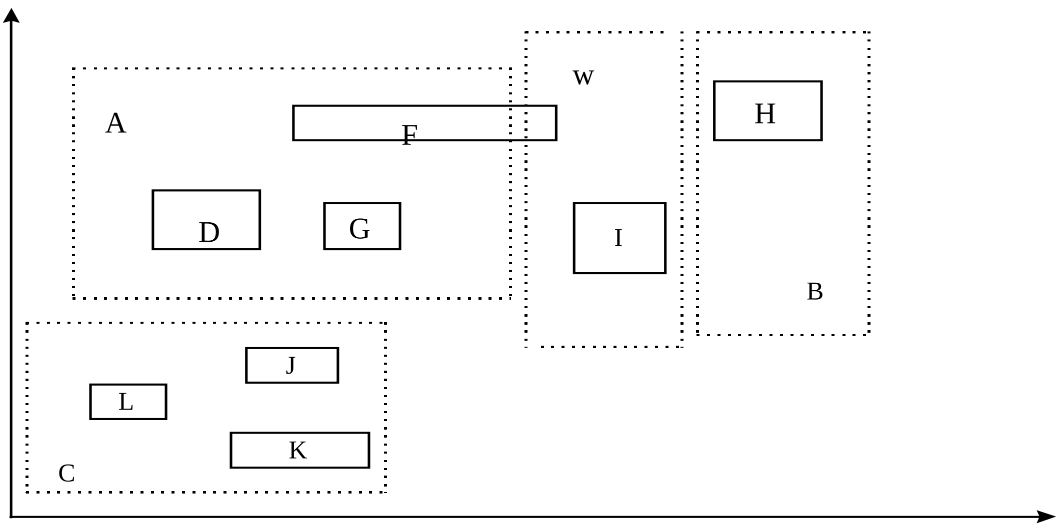
In searching, since sub-space can overlap, it is quite common that several covering rectangles stored in the same node of the tree overlap the query region. The decision to visit a subtree depends on whether the covering rectangle overlaps the query region. Therefore, the overlaps of the covering rectangle should be minimized in constructing the *R*-tree. In Roussopoulos and Leifker [1985], a packed *R*-tree has been introduced to minimize the coverage and overlap of rectangles by building an *R*-tree statically. The assumption is that for point data it is possible to partition points into groups such that the MBR of these groups do not overlap.

2.5.2.3 Object Duplication.

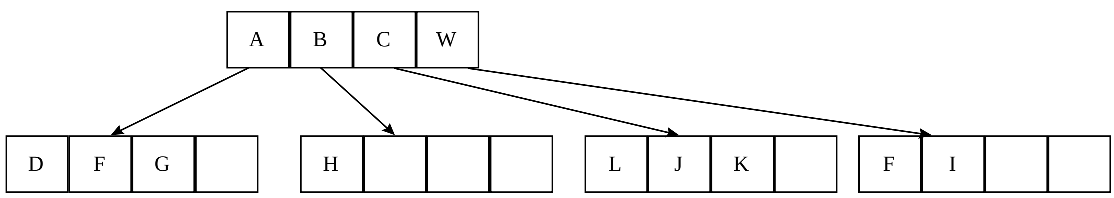
This method divides the original *n-D* space into appropriate disjoint sub-spaces. An object may be stored in multiple pages and retrieval of the object may require pruning several disjoint sub-spaces because the sub-spaces are disjoint. An example of this method is an R^+-tree that was proposed in [Sellis, *et al., 19*87]. The R^+-tree is a compromise between the *R*-tree and the *K-D-B* tree and was used to overcome the problem of overlapping covering rectangles of internal nodes of the *R*-tree. In R^+-tree, all the bounding rectangles except those leaf objects are non-overlapping, but a given rectangle may be partially covered by several bounding rectangles. Figure 2.5.2.3.1 shows the structure of R^+-tree. The difference between an *R*-tree and an R^+-tree is that nodes of an R^+-tree are not guaranteed to be at least half -filled. Also the entries of any intermediate (internal) nodes do not overlap in an R^+-tree. Moreover, an object identifier in an R^+-tree may be stored in more than one leaf node (duplication). The searching algorithm is similar to the one used in *R*-trees. The idea is to first decompose the search space into disjoint sub-regions and, for each of those, descend the tree until the actual data objects are found. The property of the duplication of MBR avoids the multiple search paths of *R*-tree for points queries and guarantees a shortest path search for all objects.

For insertion, the difference with the corresponding algorithm for *R*-tree is that multiple paths may be traversed, and the object identifier will be stored in one or more leaf nodes. The split is necessary if a leaf node is full during insertion and the split can propagate upwards to the root of the tree. The downward split may happen if the split occurs on non-leaf nodes.

Deletion of a data item from an R^+-tree can be done by first locating the data item that needs to be deleted and then removing it from the leaf node. It is similar to an *R*-tree deletion algorithm. However, since the data items may have more than one copy stored in different leaf nodes, they must be removed too. After many deletions, the storage utilization deteriorates significantly. Re-organization of the tree is required to achieve better performance.



a) Data organized in an R+-tree



b).R+tree for the rectangles of a)

Figure 2.5.2.3.1 The *R*+-tree built from Figure 2.5.2.2.1.

The *B* rectangle has been split into two sub-rectangles. Therefore all rectangles in an R^+-tree are disjoint. However, the rectangle of *F* appears in both *A* and *W*. Thus an R^+-tree introduces redundancy to solve the problem of overlapping bounding rectangle in an *R*-tree.

2.5.2.4 Summary of the Spatial Access Methods

We have reviewed some that are commonly used for indexing image databases. In brief, a linear quad-tree is proposed for indexing point objects. It supports exact and range queries and is more suitable for a static database. But the searching time increases geometrically with increased dimensions. The *kd*-tree can be used for indexing point or rectangle objects by either mapping non-zero sized objects into points in higher dimensional space or by partitioning a rectangle into smaller rectangles and storing the smaller objects in the subspace that totally includes them. The *K-D-B* tree is the combination of *kd*-tree and *B*-tree. It is height-balanced and suitable for indexing large data sets since it can page into secondary storage like *B*-tree. However both structures are not efficient in terms of searching time and storage utilization. The Grid file structure has been proposed to handle non-zero objects by mapping *k*-dimensional rectangles into higher dimensional points. Since the directory is stored as an array whose size is doubled for each directory, the storage for the grid file increases geometrically with the high dimension. Then the retrieval of objects can be costly. The *R*-tree is a generalization of a *B*-tree data structure in multi-dimensional space. It inherits the properties of the *B*-tree. Although it has the problems of overlapping, it is still a well-defined and widely used multi-dimensional indexing method. To solve the overlapping problem with *R*-tree, the R^+-tree has been proposed. In general, the R^+-tree can be more efficient than the *R*-tree. However, the performance of both structures degenerates quickly with an increased number of dimensions. In summary, existing SAM are robust for low-dimensional space but not suitable for higher-dimensional space.

2.6 Image Retrieval Systems

Image is one of the most important media data types. Huge amounts of images are being produced every day from different sources, such as scientific experiments, education, entertainment and so on. For example, about one terabyte of image data is generated by NASA'S Earth Observing System everyday [Venkat & Vijay, 1995]. To effectively retrieve images from a multimedia database, an efficient management system of the image database is required.

An image database is a database management system that allows images to be manipulated, stored, retrieved and analyzed by users in an efficient way. In general, image database systems should have the following characteristics: a) image features can be extracted based on their content. The content of image can consist of low-level features such as colour, texture and shape or of high level features such as domain specific information; b) the query can be based on an image whose features can be extracted automatically. In addition, the system should be able to handle not only exact match queries, but also similarity matched. In fact, users may not even be interested in the exact match. To support efficient image retrieval, the system should provide a suitable index on the features of the images. The issue of efficient retrieval has become more and more important in image database systems due to the tremendous sizes of images. Current image retrieval systems can be classified into two classes: text-based image and content-based image. In the following section we review the Text-Based System briefly and discuss the Content-Based Image Retrieval System in more detail.

2.6.1 Text-Based Image Retrieval System

Conventional databases have been primarily geared to the retrieval of formatted data, with the main attention focused on the identification and retrieval of structured records based on the values of certain fields. A text-based image retrieval system is based on the conventional database system by using keywords, free-text or coding to describe images. Certain keywords are chosen to index the elements of the subject of images, but these keywords are normally strictly controlled and the descriptions are manually done by human beings. Since the index fields are plain text, normal indexing methods such as inverted files or hashing can be applied directly. To answer a query, users issue some keywords, and these keywords are compared with the corresponding index field. Image database systems that use this approach include the BBC Image Library, the

Smithsonian Museums and the Image Division of the Public Archives in Canada [Leung et *al.* 1994].

The text-based approach suffers from a number of limitations. First, the descriptions of images are manually done by human beings and are rather subjective. One person's interpretation of the image may differ from another. Therefore the query could have false dismissals (*i.e.*, one or more qualifying images are not retrieved). Secondly, it is not natural and is incompatible with human visual systems. It fails to capture the rich content of the images, since all the visual properties are not possibly be described with a few keywords or free-text. Thirdly, this approach tends to address an individual element of an image in isolation so that it can not reflect the full semantic meaning embedded in the image. Finally, the manual description of the image is laborious and is limited to a small-scale image database.

2.6.2 Content-Based Image Retrieval System

To overcome the limitations of text-based image retrieval systems, the Content-Based Image Retrieval system is proposed. It has attracted much attention of researchers recently. Content-based systems are based on an understanding of the low-level content of the image in image databases. The system requires the combination of image processing and database system management techniques. In such systems, image contents are extracted as image features by applying various image-processing techniques. To query an image database, a complex user interface is required. Users can construct a query by picking a colour from a colour palette, or selecting texture from a menu of texture patterns or making free-hand sketch on the screen. In order to retrieve the image from the image database efficiently, a suitable access method is required. Unlike the existing database management systems, the index of an image database is constructed on the corresponding feature vectors files of high dimension. To answer users' queries, the index files are searched and qualified results with images are retrieved and presented to users. Figure 2.6.2.1 shows the architecture of the CBIR system.

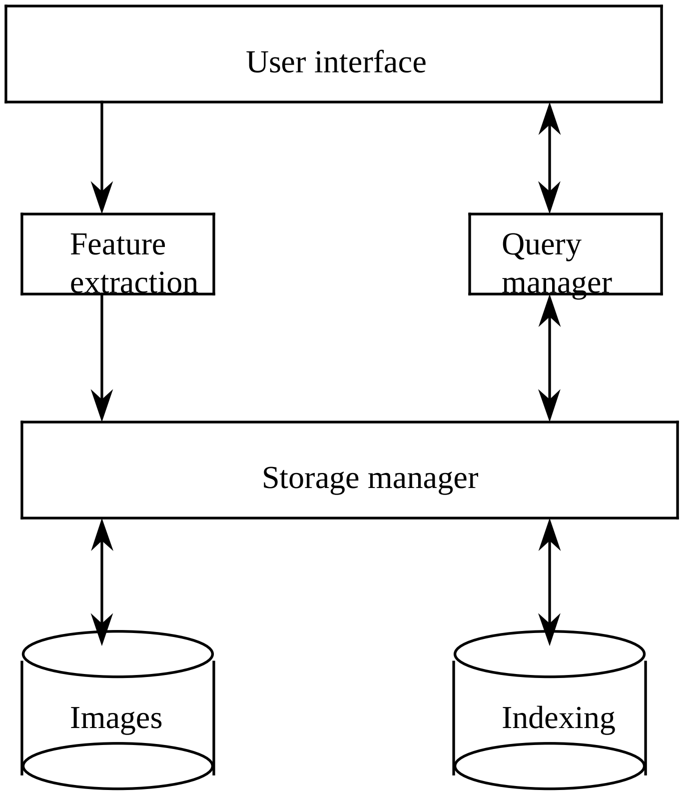


Figure 2.6.2.1 The architecture of the CBIR system

2.6.3 Examples of Content-Based Image Retrieval Systems

Many of the earlier image retrieval systems are attended on text-based retrieval and have very limited capabilities for retrieving images based on human visual perceptions. Over recent years, research has focused on the use of internal attributes of images and videos computed in an automatic or semi-automatic way. Many CBIR systems are being built in recent years such as QBIC, Photobook, Virage, JACOB, SWIM, and CORE. In what follows, we give a brief review of some of the existing systems.

The QBIC system (Query by Image Content) [Niblack *et al.*, 1993] was developed by IBM, at Almaden, USA, and is based on a priori feature extraction approach. The QBIC system allows queries of large image databases based on visual image content properties such as color percentages, color layout, and textures. Such queries use the visual properties of images, so users can match colors, textures and their positions without describing them in words. This system also combines content-based queries with keywords and text predicates to get powerful retrieval methods for image and multimedia databases.

Two major properties of QBIC are its use of image content-computable properties of colour, shape, and textures; and its graphical query language. To utilize these properties, QBIC has two main components: one is the database population which is concerned with the process of object identification and extracting the features of image. The other component allows users to compose a query graphically. Features are generated from the graphical query and then input to a matching engine that finds images from the database with similar features. The similar queries are done based on the database of pre-extracted features using the distance function between features.

In order to achieve fast searching matching, a multidimensional indexing method, R*-tree was used in QBIC for low-dimensional single features such as color and texture (each 3D). For high-dimensional single features, such as 20-dimensional moment-based shape feature vectors, K-L or principal component linear transformation was used to reduce the features dimensionality. Then an R*-tree was adapted to index the image database.

Another example of a CBIR system is the photobook project [Pentland, *et al*,.1994] developed at the Media Lab of MIT. This system addresses the issue of predetermined search criteria by storing enough information about each image. It works by comparing features associated with images, not the images themselves. These features are in turn the parameter values of particular models fitted to each image. These models are commonly color, texture, and shape, though Photobook claims that it is capable of working with features from any model. Searches can be based on image shape, texture and shape features. Photobook provides a library of matching algorithms for comparing the image features. In this system, images are compressed by encoding according to category. The matching algorithms include Euclidean, Mahalanobis, Divergence, Vector space, Histogram, Fourier peak, and Wavelet tree distances, as well as any linear combination of these. Users can compose new matching algorithms based on the above matching algorithms by dynamic code loading.

Background and Related Work

Several functions are provided by Photobook such as texture modeling, face recognition, shape matching, brain matching, interactive segmentation and annotation. It also includes FourEyes, an interactive learning agent which selects and combines models based on examples from the user. This makes Photobook different from tools like QBIC, Virage, SWIM, and CORE, which all support searches on various features but offer little assistance in actually choosing one for a given task. FourEyes, by contrast, allows users to directly address their intent.

The JACOB system, allowing content-based annotation and querying in video databases, has been developed at the Computer Science & Artificial Intelligence Laboratory of the University of Palermo, USA. JACOB has an architecture that can be subdivided into two functional units: the first one is dedicated to the database population and the second one to the database querying. In the database population unit, images and videos are characterized in the usual way. Two different archives are managed, the one containing raw data, the other containing features. During a database population session a new video or image is added to the *DB*, the system automatically splits a video into a sequence of shots, extracts a few representative frames (called r-frames) from each shot and computes r-frame descriptors based on color, texture and motion. Thus no user action is required during the database population step. However it is suggested that some manual annotation is still essential if the application requires retrieval to be related also to traditional features, e.g., object names, dates, and so on. The user interaction is necessary only during the querying phase.

During a database querying session user performs a query to retrieve the desired image from the DB, typically the user works in an iterative way. Queries may be posed as a direct query or through an example. To perform a direct query, the user inserts a few values representing the color histogram and/or the texture features and/or the motion-based features. Queries (direct or by example) are retrieved by a sequential scan of the feature DB; similarity among images is evaluated by using different metrics for different features. The user establishes the importance of different features by taking into account the statistics of the database and the weights provided. Queries may be based on one or more of the above-mentioned features and the user can specify the relative weight of various features.

When performing multi-feature queries, the user chooses a value between 0 and 1 to assess the importance of a feature with respect to another one. In order to obtain a global similarity value for each feature descriptor (color, texture, and motion), a method to adequately weight the distance computed is needed. Otherwise, images with a similar color distribution could give a similarity value different from the one obtained from images with similar textures. The same holds when other features are considered. This method is sometimes called distance normalization and is based on a statistical analysis of feature distances. The system computes a probability distribution for the metric which associated with each feature, comparing every r-frame in the database (about 1500) with all others. Thus a probability distribution for the dissimilarity measure related to each feature is obtained. These probability distributions are used to make all the distances fit the range [0,1]. This approach makes the underlying assumption that images currently contained in the database are representative of all images that the system could ever contain.

The major characteristics of the JACOB system are the automatic feature extraction capability (no human action is required during the DB population step) and the domain independence. However, image retrieval of this system is based on sequential scans and the similarity measure of images is based on a probability distribution which may restrict its application.

In summary, many CBIR systems have been developed to address different aspects of the problems. At present, they are all still prototype systems, but the technology from these prototype systems has quickly moved into commercial stand-alone products. For example, IBM’s Ultimedia Manager is part of IBM’s Digital Library and the DB2 series of products include the technology from QBIC systems. Virage's VIR (Visual Information Retrieval) technology has been licensed to Oracle Corporation which plans to distribute Virage's VIR image engine as a data cartridge that plugs in to the Oracle Universal Server.

Chapter 3

Linear Dimension Reduction

3.1 Introduction

In any image system, image features which are extracted by different image processing techniques typically involve high-dimensional feature vectors because of the large numbers of model parameters used to model images. Some parameters in these models are redundant for content-based retrieval, but it is very hard to discover these redundancies at the image processing stage. Therefore, it is necessary to use dimension reduction techniques to eliminate the redundancy of image features after the image processing stage. A sufficiently low-dimensional model of the feature vectors can lead to optimal retrieval performances in image databases.

The process of feature dimension reduction transforms a high dimensional feature space into a low-dimensional feature space without significant loss of the image properties. The goal of the feature dimension reduction is to discover complex dependencies among the features of images, eliminate information redundancy or disturbing noise and maintain sufficient information to discriminate between image classes. As a result, information storage, transmission, retrieval and processing can become more efficient due to the low dimensionality of data.

Many dimension reduction methods have been proposed which can be broadly classified into two categories: linear dimension reduction (LDR) and nonlinear dimension reduction (NLDR). In general, the main difference between LDR and NLDR is that NLDR can enable the system to maintain a great deal of knowledge about the information on the data source reflected in the data samples. This information can be represented in network weights between units in successive layers of the network. Thus

Linear Dimension Reduction

NLDR can be used for very difficult tasks which can not be handled by LDR. However, training networks for NLDR may be slow. On the other hand, the LDR computation can be fast. LDR is discussed in this chapter and NLDR is discussed in Chapter 4.

Linear dimension reduction is well known as an effective process of mapping the linear original features into more effective features by eliminating the redundancy among original feature spaces. The mapping function is well defined. It is simple to find the coefficients of linear functions so as to maximize or minimize a criterion. Although, in many image applications, there are important features which are not linear functions of the original measurements, LDR is still commonly chosen for feature dimension reduction.

The idea of LDR is to rotate the *n*-dimensional coordinate system to align the data, and then choose *m* ($m < n$) dimensions that can best account for the variance. One particular form of constraint that can occur over a collection of feature data points is shown in Figure 3.1.1. It assumes the feature space has two dimensions and each feature point is represented by its (x, y) coordinates. Under some constraints, all the feature points can appear to lie on a one-dimensional curve. If one can discover the one-dimensional curve, then all two-dimensional feature data samples can be represented by a single number alone on the constrained curve [Saund, 1980].

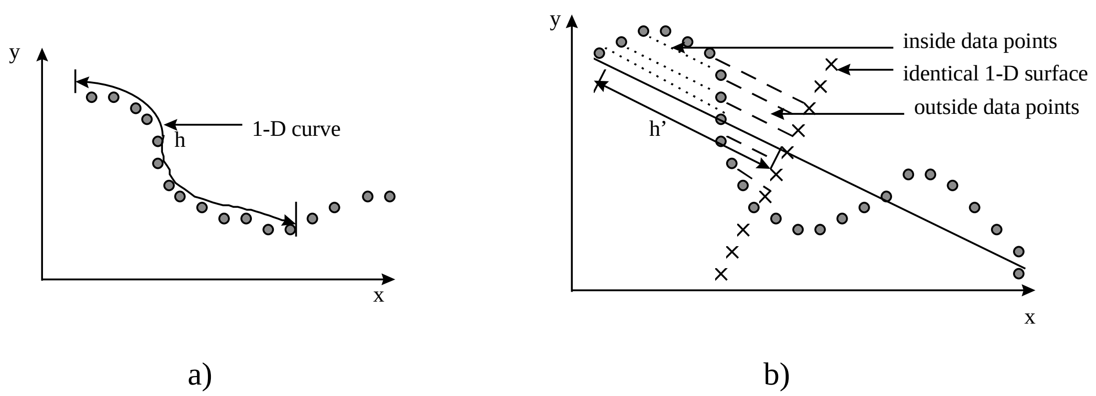


Figure 3.1.1 2-D data samples constrained to lie on a one-dimensional constrained surface.

Linear Dimension Reduction

There are two cases in Figure 3.1.1: a) through dimensionality-reduction, the parameter h describes data in terms of location on the constraint surface; b) through the data inside (closed circles) and outside (open circles) which is described by the parameter h', the constraint surface appears identical under dimensionality-reduction.

Since LDR assumes a linear constraint surface, many classical linear statistical techniques can be used for LDR such as factor analysis and principal components analysis [Fukunaga & Koontz, 1970, Kittler & Young, 1973, Watanabe, 1965]. These techniques require less computation since the calculation is based on matrix transformation which is well defined in pure mathematical theory. Based on these properties, LDR has been the subject of intense study in recent years. For example, Ljubomir [1993] proposed LDR based on the Bays-optimal technique for pattern recognition. This method is to find a transformation matrix that would minimize the Bays error in the low-dimensional feature space and compute the raw data set by the projection on the low-dimensional space.

Flemming and Adrian [1995] used LDR (factor analysis) for minimizing noise in linear dimension reductions of sequences of medical images based on the given factors, such as the covariance matrix and the mean of noise. The LDR is used in two-step processes: first, the factors (signal spaces) are calculated and then the actual dimension reduction of the sequence of images is performed. However, this work focuses on a single pixel within the sequence of medical images and tries to remove the noise from individual pixels. There is no optimal weighting procedure for this approach. Instead, an optimal metric for a specific factor analysis model is used.

3.2 Overview of Principal Component Analysis

Principal Component Analysis is one of the most general forms of LDR. It extracts the most important features by finding the directions in input space in which the inputs have the most variation. Then the n-dimensional input space is projected onto the m-dimensional linear subspace to achieve dimension reduction ($n < m$). In the subspace, these directions are in fact spanned by the eigenvectors of the correlation or covariance matrix of the data corresponding to the m largest eigenvalues [Watanabe, 1970]. The m

value is defined by specifying the tolerable mean square error in reduced representation. The PCA is based on second-order correlation. It has been employed to reduce the image feature dimension so that an efficient index can be constructed for image retrieval in image databases [Lee *et al.,* 1993, Euripides & Faloutsos, 1989]. It can also be used for image coding, *i.e*., for removing correlations from highly correlated data such as face images [Sirovich & Kirby, 1987]. The following sections will give some basic ideas of LDR and discuss PCA in detail.

The central idea of PCA is to reduce the dimensionality of a data set which may consist of a large number of interrelated variables, and retain as much as possible of the variation present in the data set. This is achieved by transforming the original data set into a new set of variables namely principal component (PCs), which are uncorrelated and are ordered so that the first few retain most of the variation present in all of the original variables [Jolliffe, 1986].

The earliest descriptions of PCA techniques have been given by Pearson [1901] and Hotelling [1933], using different approaches. Hotelling introduced the standard algebraic derivation, while Pearson [1901] was concerned with finding lines and planes which best fitted a set of points in *n*-dimensional space, and with the ability to include geometric optimization in the PCA. Pearson stated that his approach "can be easily applied to numerical problems, although the calculation may become cumbersome for four or more variables." It is quite interesting since these comments occurred over 50 years before the widespread availability of computers.

Hotelling's approach starts from the idea of factor analysis but is rather different because he assumes that there may be a smaller fundamental set of independent variables that determine the values of the original set of *p* variables. Hotelling regards these fundamental sets of variables as the "principal components." The analysis that finds such components is then called the "method of principal components."

The derivation of PCs in Hotelling's approach is similar to factor analysis, using Lagrange multipliers and ending with an eigenvalue and eigenvector problem. However, there are some differences from the factor analysis. First, Hotelling's approach works with a correlation, rather than a covariance matrix; second, his approach focuses on the original variables expressed as linear functions of the components rather than on the components expressed in terms of the original variables; finally, his approach does not use matrix notation.

Although Girshick [1939] also investigated the variances of PCs, there appears to have been relatively little work on the development of different applications of PCA during the 25 years immediately following the publication of Hotelling's paper. However, with the widespread introduction of electronic computers and the explosive development of new applications, further theoretical work on PCA has occurred.

Despite the apparent simplicity of the PCA, there is much research still needed to be done on both the general and applications areas of PCA. PCA can be applied to a variety of discipline such as agriculture, biology, chemistry, climatology, demography, ecology, economics, food research, geology, meteorology, psychology, quality control and computer science.

3.3 Definitions of Principle Components

For q random variables, unless q is small or the structure is very simple, it is not easy to simply look at the q variances and all of the $1/2q\ (q\ -\ 1)$ correlation or covariance. Instead, another approach is required to look for a few (<< q) derived variables that preserve most of the information given by these variances rather than the correlation or covariance. PCA is such an approach to achieve this transformation because it concentrates on variance, rather than covariance and correlation.

Supposing that x is a vector of q random variables, the first step of PCA is to find a vector $\beta_1 \ni \Re^q$ such that the linear function $\beta_1^T x_j$, $j = 1 \ldots N$ have maximum variance, where

$$\beta_1^T x = \beta_{11}x_1 + \beta_{12}x_2 + \ldots + \beta_{1q}x_q = \sum_{j=1}^{q} \beta_{1j}x_j \qquad (3.3.1)$$

Then, look for another vector β_2 that is linearly independent to β_1 and that gives maximum variance for $\beta_2^T x_j$, j =1…N, and so on. In summary, at the i-th step, vector β_I is determined such that it is linearly independent to β_1, β_2,..β_{i-1} and the variance of { $\beta_i^T x_j$ | j =1…N } is maximum. The scalar $\beta_i^T x$ is called the i-th PC. In general, there are q maximum PCs that can be derived; however, it is desirable to have the first n PCs, where $n << q$, to account for the most of variation of the original data.

3.4 Derivation of Principal Components

Mathematically, this method can be defined as follows: Given a data matrix X ϶ $\Re^{n\times N}$ where the j-th column of X is x_j ϶ $\Re^n$ representing the j-th observation vector, and a set of linearly independent basis vectors {q_i|i = 1… N}, then each of these x_j's can be expressed as a linear combination of the basis vectors, i.e.

$$x_i = \sum_{1=1}^{n} y_{ji}\, \mathbf{q}_i = Q\mathbf{y}_j, \tag{3.4.1}$$

Where

$$Q = [\mathbf{q}_1, ..., \mathbf{q}_n] \ni \Re^{n\times N}$$

And

$$\mathbf{y}_j = [y_{j1}, y_{j1}, ..., y_{jn}]^T \ni x_j \ni \Re^n$$

Since Q consists of linearly independent columns

$$|Q| \neq 0$$

The columns of Q span the n-dimensional space containing each x_j. If the orthonormality condition is satisfied, i.e.

$$q_i^T q_k = \begin{cases} 1 & if\ i = k \\ 0 & otherwise \end{cases} \tag{3.4.2}$$

$$||q_i|| = 1 \quad \forall i,$$

then y_i can be expressed in terms of Q and $\mathbf{x}_j$ as follows

$$\mathbf{y}_j = Q^T\mathbf{x}_j \qquad (j = 1, ..., n) \tag{3.4.3}$$

i.e. the i-th components of $\mathbf{y}_j$ is simply:

$$y_{ji} = q_i^T \mathbf{x}_j$$

The quantity y_{ji} is called the i-th component of the observation vector $\mathbf{x}_j$ in the mapped space of the feature vector q_i:

The sample mean vector $\bar{x}$ of the observation data vectors is defined as

$$\bar{x} = [\ \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_N\ ]^T = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad (3.4.4)$$

The mean $\bar{x}_j$ of the kth component is given:

$$\bar{x}_k = \frac{1}{N}\sum_{i=1}^{N} x_{ik} \qquad (3.4.5)$$

and the difference vector d_j between the mean and each observation vector is simple

$$\mathbf{d}_j = \mathbf{x}_j - \bar{x} = \begin{bmatrix} x_{j1} - \bar{x}_1 \\ x_{j2} - \bar{x}_2 \\ \ldots \\ \ldots \\ \ldots \\ x_{jn} - \bar{x}_n \end{bmatrix} \qquad (3.4.6)$$

Linear Dimension Reduction

Let

$$X_c = [d_1, d_2, \ldots, d_n] = \begin{bmatrix} x_{11} - \bar{x}_{1,} & \ldots, & x_{N1} - \bar{x}_1 \\ x_{12} - \bar{x}_{2,} & \ldots, & x_{N2} - \bar{x}_{2,} \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ x_{1n} - \bar{x}_{n,} & \ldots, & x_{Nn} - \bar{x}_{n,} \end{bmatrix} \tag{3.4.7}$$

Then the covariance matrix is defined as $S = X_c X_c^T = [s_{ij}]$, where

$$\begin{aligned} s_{ij} &= \sum_{k=1}^{N} (x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j) \\ &= \sum_{k=1}^{N} x_{ki} x_{kj} - N\bar{x}_i \bar{x}_j, \quad j = 1, \ldots, n \end{aligned} \tag{3.4.8}$$

Define the transformation Γ: $\Re^n \rightarrow \Re^n$ as

$$y = \Gamma(\mathbf{x}) \equiv Q^T (x - \bar{x})$$

then the original of the transformed data is now at the mean value $\bar{x}$ (i.e. $\bar{y} = 0$) and the coordinate axes coincide with the principal component vectors $q_1, q_2, \ldots, q_n$.

If the original observation data vectors span only space $\Re^d \subset \Re^n$ then the first d eigenvectors $q_1, q_2, \ldots, q_n$ are sufficient for representing the sub-space $\Re^d$. This choice of the origin guarantees that the sum of squared errors of the projections of x_j, $j = 1 \ldots N$ onto the sub-space is minimized. Given $Q_1, Q_2, \ldots, Q_d$ as orthonormal vectors for the d-dimensional axes, the sum of projection is:

$$\sum_{i=1}^{n} s_{ij} - \sum_{i=1}^{d} Q_i^T S Q_i \tag{3.4.9}$$

To minimize (3.4.9), the value of $\sum_{i=1}^{d} Q_i^T S Q_i$ must be maximum. Subject to $Q_i^T Q_i = 1$, using the technique of Lagrange multipliers and maximize

$$Q_i^T S Q_i - \lambda_i (Q_i^T Q_i - 1), \qquad (3.4.10)$$

where λ_i is a Lagrange multiplier. The differentiation with respect to Q_i^T gives

$$\Sigma Q_i - \lambda_i Q_i = 0, \qquad (3.4.11)$$

or

$$(\Sigma - \lambda_I I_p) Q_i = 0, \qquad (3.4.12)$$

where I_p is the $(p \times p)$ identity matrix. Thus, λ_i is an eigenvalue of S and Q_i is the corresponding eigenvector. To decide which of the p eigenvectors is the maximizing value of Q_i, the following equation is used:

$$Q_i^T SQ_i = Q_i^T \lambda_i Q_i = \lambda_i Q_i^T Q_i = \lambda_i, \qquad (3.4.13)$$

so λ_i must be as large as possible. Thus, Q_i is the eigenvector corresponding to the largest eigenvalue of S, and $Q_i^T SQ_i = \lambda_i$, is the largest eigenvalue.

3.5 Computation of Principal Components

Finding Principal Components can be reduced to finding the eigenvalues and eigenvectors of a positive semi-definite matrix. The type of algorithm used for PCA is important, especially if the dimension (d) of the target feature space is much lower than the original feature space (n) ($d << n$). The power method and the singular value decomposition are the most common algorithms for computation of PCs.

3.5.1 The Power Method

The power method is a technique for finding the largest eigenvalue and the corresponding eigenvector of a $(n \times n)$ matrix T. The idea is to choose an initial n-element vector v_0, and then form the sequence:

$v_1 = T\, v_0,$

$v_2 = T\, v_1 = T^2\, v_0,$

.   .   .

.   .   .

.   .   .

$v_i = T v_{i-1} = T^i\, v_0,$

.   .   .

.   .   .

.   .   .

$$v_1 = T\, v_0 = \sum_{i=1}^{n} k_i\, \varphi_i, = \sum_{i=1}^{n} k_i\, \lambda_i \varphi_i, \qquad (3.5.1.1)$$

where λ_1, λ_2, ..., λ_n are the eigenvalues of T. In general for $r = 2, 3, ...,$

If the eigenvectors of T are defined as φ_1, φ_2, ..., φ_i, then they form a basic for *n*-dimensional spaces, and for an arbitrary v_0,

$$v_0 = \sum_{i=1}^{n} k_i\, \varphi_i, \qquad (3.5.1.2)$$

where the k_i are some set of constants. Then

$$v_r = \sum_{i=1}^{n} ki\, \lambda_i^r \varphi_i, \qquad (3.5.1.3)$$

and

$$\frac{v_r}{k_1 \lambda_1^r} = (\varphi_1 + \frac{k_2}{k_1}(\frac{\lambda_2}{\lambda_1})^r \varphi_2 + ... + \frac{k_n}{k_1}(\frac{\lambda_n}{\lambda_1})^r \varphi_n). \qquad (3.5.1.4)$$

If the first eigenvalues of T are distinct from the remaining eigenvalues, so that $\lambda_1 > \lambda_2 \geq ... \geq \lambda_n$, then it follows that a suitably normalized version of $v_r \rightarrow \varphi_1$, as $r \rightarrow \infty$.

The power method thus gives a simple algorithm to finding the first PC and its variance and can be adapted to find the second, third and the rest of the PCs. However, this method will converge very slowly if λ_1 is not much larger than λ_2. There are various modified power methods which can partially overcome the problem just mentioned. Many of them can be found in Wilkinson [1965].

3.5.2 Singular Value Decomposition

Singular value decomposition (SVD) is another importance technique for the computation of PCs. Given an arbitrary matrix X of dimension $n \times m$, it can be assumed that X will invariably be a matrix of n observations on m variables, measured about the means, X can be written

$$X = VUA^T \qquad (3.5.2.1)$$

where

(1) V, A are $(n \times r)$, $(m \times r)$ matrices respectively and each of them has orthonormal columns, that is $V^TV = I_r$, $A^TA = I_r$;

(2) U is a $(r \times r)$ diagonal matrix;

(3) r is the rank of X.

For each element of X,

$$x_{ij} = \sum_{k=1}^{r} v_{ik} u_k^{1/2} a_{jk}, \qquad (3.5.2.2)$$

where v_{ik}, a_{jk}, are the $(i, k)th$, $(j, k)th$ elements of V, and A respectively and $u_k^{1/2}$ is the kth element of U. Thus x_{ij} can be split into parts:

$$v_{ik} u_k^{1/2} a_{jk}, \qquad k = 1, 2, ..., r,$$

which are related to each of the first r PCs. If only the first p PCs are retained, then

$$ {}_p x_{ij} = \sum_{k=1}^{r} v_{ik}\, u_k^{1/2}\, a_{jk}, \qquad (3.5.2.3) $$

will provide an approximation for x_{ij}. Therefore the SVD provides a sequence of approximations, of rank *1, 2, ..., r* to X which minimizes the Euclidean norm of the difference between X and the approximation ${}_pX$.

It has been suggested that PCs may best be computed using the singular value decomposition technique (SVD) [Chambers, 1977, Gnanadesikan, 1977]. The importance of the SVD for PCA is twofold. First, it provides a computationally efficient method of actually finding PCs. The second virtue of the SVD is that it provides additional insight into what a PCA actually does, and gives useful means, both graphical and algebraic, of representing the result of a PCA. Finally, if PCA is used in conjunction with regression, the SVD can be useful for regression analysis [Mandel, 1982; Nelder, 1985]. We use SVD algorithms for our PCA in this thesis.

3.6 PCA and Feature Dimension Reduction

The main purpose of this section is to give an example illustrating how the PCA can be used for image feature dimension reduction. The data for this example consist of 30 texture feature variables, which are derived from nine images from Appendix 1. The first step when applying PCA to a data set is to obtain the correlation matrix of the data set, together with the standard deviations of each of the 30 feature variables, which are also given in Appendix 1. Then we can get the eigenvalues and the percentage of the total variation explained for this correlation matrix. Table 3.6.1shows the eigenvalues and the percentage of the total variations.

Table 3.6.1 The eigenvalues and the percentage of total variation

	1	2	3	4	5	6	7	8	9	10	11-30
λ	11.529	6.800	3.222	2.760	1.523	1.174	1.00	1.00	0.617	0.372	0.000
%	38.4	22.7	10.7	9.2	5.0	0.33	0.31	0.31	2.0	1.2	0.0

The eigenvalues are in descending order, *i.e.*, $\lambda_1 > \lambda_2 > \ldots > \lambda_{30}$. The first five eigenvalues explain *95.85%* of the total variance and should be sufficient to analyze the influence of the image feature. If we choose the first five eigenvalues and calculate the corresponding five PCs, then the *30*-dimensional of image feature data set can be reduced to a *5*-dimensional feature data set. Later, we will discuss how many PCs we really need to best represent the original data set. The eigenvectors for the nine images are given in Table 3.6.2. The corresponding five PCs are shown Table in 3.6. 3.

Table 3.6.2 The first five eigenvectors

λ_1	λ_2	λ_3	λ_4	λ_5
-0.0584	-0.1347	0.4201	0.0935	-0.2498
-0.0288	-0.2108	-0.1998	-0.3824	-0.2650
0.0163	-0.1967	-0.3183	-0.1971	-0.2888
-0.0148	-0.2264	-0.3227	-0.1046	-0.2736
0.0610	-0.2547	0.3233	0.1958	0.0330
0.1593	-0.1377	-0.3635	0.1566	0.0326
0.2586	0.0166	-0.0660	-0.1957	0.2219
0.0837	-0.2639	0.0789	0.097	-0.2243
0.0682	-0.3064	0.0459	0.2627	-0.0174
0.2082	-0.0505	-0.1262	-0.3125	0.2270
0.2137	-0.2046	-0.0993	0.1283	0.1857
0.2472	0.0701	0.1911	-0.1427	0.2246
0.2386	-0.2076	0.0640	0.0479	-0.0045
0.1854	-0.1286	0.0704	0.2929	-0.0187
0.2454	0.0879	0.0575	-0.2304	0.2445
0.1720	0.0437	-0.2832	0.2607	-0.1158
0.2854	0.0320	0.0064	-0.085	0.0428
0.2640	-0.0455	-0.1082	0.2062	-0.0200
0.2761	-0.0164	-0.0975	0.1293	0.0983
0.2907	0.0225	0.0452	-0.0261	-0.0183
0.1749	0.2321	0.0882	-0.0403	-0.2172
0.1849	0.1249	0.1823	-0.1905	-0.4289
0.2751	-0.0479	0.0362	0.0722	-0.1949
0.2290	0.1618	0.1698	-0.0138	-0.2552
0.2257	0.1794	0.0932	-0.1771	-0.1763
-0.0158	-0.3402	0.1536	-0.2084	0.0807
0.0000	-0.0005	0.0003	-0.0004	0.0003
-0.0158	-0.3402	0.1536	-0.2084	0.0807
-0.0158	-0.3402	0.1536	-0.2084	0.0807
0.0000	-0.0005	0.0003	-0.0004	0.0003

Table 3.6.3 The first five PCs

Image	PC1	PC2	PC3	PC4	PC5
Image1	-0.7821	0.0188	-0.0917	-0.4592	0.2750
Image2	-1.1182	0.4990	-0.2725	0.3476	0.4538
Image3	-1.1656	-0.2874	0.3437	-0.4317	0.3846
Image4	0.3556	-1.3189	1.2533	0.0882	-0.1402
Image5	-0.1050	-0.9138	-0.9621	-0.7486	-0.5597
Image6	1.9045	-0.8546	-0.6021	0.7332	0.3490
Image7	-0.2079	0.8254	0.2413	0.4744	-0.7826
Image8	1.9514	1.4732	0.2692	-0.6770	0.1261
Image9	-0.8327	0.5583	-0.1790	0.6730	-0.1060

The first five PCs account for *95.85%* of the total variance among the nine image features data. To better illustrate the interpretation of the PCA for this data set, we select the first two PCs and plot the data set with respect to these two dimensions. This gives a straightforward visual representation of what the data look like, instead of having to digest a large mass of numbers. The first two PCs for the above data set account for *38.4%* and *22.7%* of the total variation respectively. Thus, a two-dimensional plot with respect to these PCs, representing *61.1%* of the variation, will give a reasonably good approximation to the relative positions of the observations in the *30*-dimensional space. Plotting this data set with respect to the first two PCs is illustrated in Figure 3.6.1

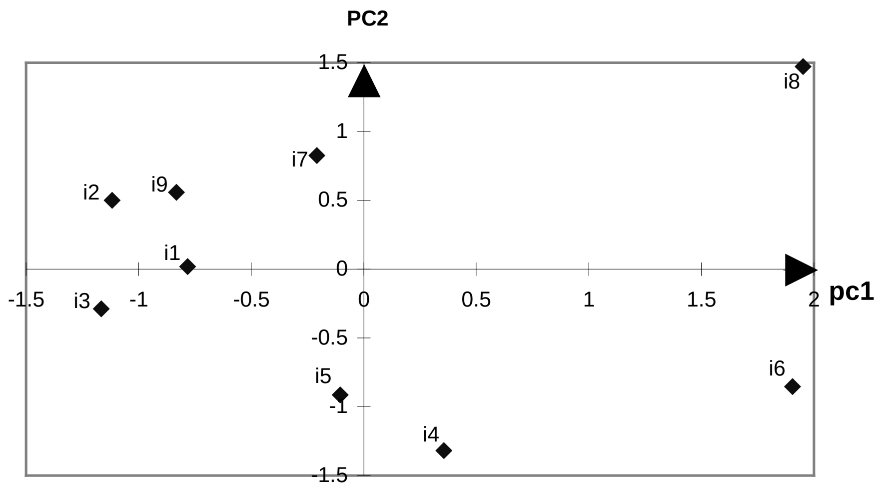


Figure 3.6.1 Plot of *9* image features with respect to their first two PCs

It was noted in the *2*-dimensional plot that the total variation of the nine observations of the *30*-dimensional space is *61.1%*. The percentage of the total variation is an obvious measure of how good the *2*- dimensional representation is.

Since PCA is a linear transformation method, applying PCA directly to a nonlinear data set such as the image features set can result in a wrong representation of the original data set in a lower dimensional space. In the above example, since the nine images are grouped into three clusters (image 1-3; image 4-6; image 7-9) based on human perceptions, we expected the three groups to be observed after transformation. However, in Figure 3.6.1 it can be seen that the observations of data points do not reveal these three clusters. The three groups of images overlap intensely in the *2*-dimensional space. For example, observations *1*, *2* and *3*, which were classified into one group based on human visual perceptions, are no longer in a group. Observations *9* and *2*, which are not in the same group, now are closer than Observations *2* and *3*. Thus no distance function can be used to represent their similarity based on the original image. In other words, the dimension reduction of image features based on this method may not yield good results for maintaining the structure of the original image data set.

3.7 Discussion

3.7.1 Principal Component Selected

The major objective in many applications of PCA is to reduce *n*-dimension V vector by replacing the n elements of V with a much smaller number, *m* of PCs, which discard very little information. It is necessary to know how small *m* can be taken without seriously losing information. Several rules are available for selecting suitable *m* in such a way that most of the information available from the original V vectors is retained. In this section, we present one of the rules, Cumulative Percentage of Total Variation, for deciding how many PCs should be retained in order to account for most of the variation in V.

This rule is to select a (cumulative) percentage of the total variation which the selected PCs should contribute, say *70%* or *80%*. The required number of PCs is then the smallest value of m for which this chosen percentage is exceeded. Principal components are successively chosen to have the largest possible variance and the variance of ith PC is q_i. Thus the sum of the variances of PCs is equal to the sum of the variances of the variable of V. Therefore for the first m PCs, the percentage of variation accounted for is defined as:

$$P_k = 100 \sum_{j=1}^{m} q_j \Big/ \sum_{j=1}^{n} q_j \qquad (3.7.1.1)$$

In the case of a correlation matrix of table 3.6.1 this is reduced to

$$P_k = \frac{100}{n} \sum_{j=1}^{k} q_j \qquad (3.7.1.2)$$

In order to define the value m, where m is the smallest integer, a cut-off threshold P_k^* between 60% and 90% is chosen so that $P_k > P_k^*$. This rule provides a guideline to preserve most of the information in the first m PCs.

In fact, using this rule is equivalent to looking at a spectral decomposition of the SVD of the data matrix that was described already. In practice, the eigenvalues can be used to decide the value m. For example, in Table 3.6.1, if we choose $m = 5$, the total percentage of variation covered by the first five PCs is *95.85%* which is sufficient to preserve most of the information in the original vector. In a later chapter, we will use 95% as the cut-off value to determine the number of PCs required.

3.7.2 PCA for Compositional Data

Compositional data consists of observations $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n$, for which each element $\mathbf{z}_i$ is a part of the overall observations. For example, when a number of image features such as colour, texture and shape, are analyzed together, it is called compositional data. The vectors of colour, texture and shape features become a proportion of the compositional data vector. Each element of the compositional data has different constraints and often exhibits high non-linear data structures among their variables. It is not suitable to apply PCA directly to these compositional data. Aitchison [1983] has proposed a modified the PCA for such data. The idea of this modified PCA is to do a PCA on a subset (n - 1) of the compositional variable, but this idea has an unsatisfactory feature because the choice of the variable to leave out is arbitrary. Different choices lead to different PCs. For example, suppose two variables have much larger variances than the other (n - 2) variables, then the result of the PCA for the (n - 2) covariance matrix will vary considerably, which may depend on whether the omitted variable has a large or small variance. The modified PCA does not make much difference to the result of a PCA.

In section 3.6, an image data set with texture features only for nine images was discussed. Instead of using texture features alone, we discuss the data set that includes texture as well as colour. The data now becomes compositional in nature which is a *31*-dimensional data set (texture: *30D*; colour: *1D*). A modified PCA was carried out on these compositional data. Figure 3.7.2.1 gives the result of variances for the first two PCs for the modified PCA. It can be seen that the PCs in Figure 3.7.2.1 and Figure 3.6.1 are very similar.

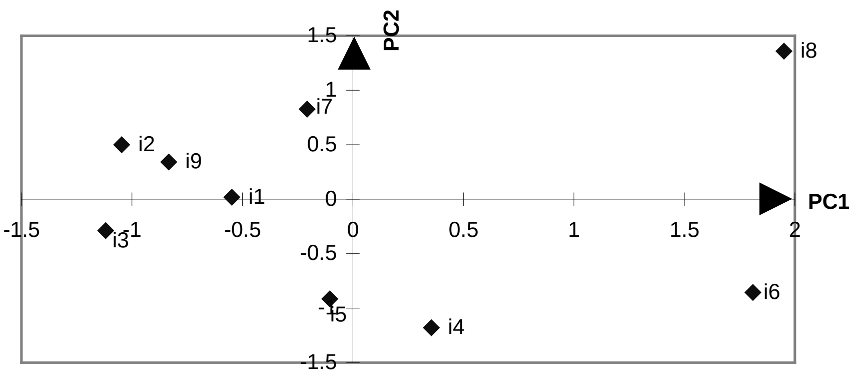


Figure 3.7.2.1 Plot of 9 images with texture and colour features in 2-D space.

3.8 Concluding Remarks

PCA is a widely used statistical technique for data analysis. It is fast and is particularly good for recovering the linear data structure in a lower dimension feature space. An overview of PCA was presented in this chapter. We presented the formal mathematical definitions of Principal Component and discussed the theory for the derivation of the PC. Then the algorithms for calculating the PCs were given. Several properties of a PCA which relate to image dimension reduction were also discussed.

An example of using the PCA for reducing the dimension of an image feature was also given in this chapter. In this example, the PCA is used as a stand-alone method. The result in Figure 3.6.1 shows that the linear transformation of the PCA is insufficient for distinguishing image similarity for individual features or for combining multiple images features. However, a PCA could be used as the first step for the NLDR where it will provide optimal first pass reduction for the removal of covariances and speed up the NLDR training time. In our work, we use a PCA as a first step to the dimension reduction of image features and feed the results to the NLDR methods for further reduction.

Chapter 4

Non-Linear Dimension Reduction

4.1 Introduction

Linear dimension reduction assumes the data lie on a linear constrained surface and does not work for non-linear constrained surfaces. It can not accurately determine if a given, unknown data sample does or does not lie on the non-linear surface when projecting data onto a low-dimensional non-linear subspace. Projecting a non-linear n-dimensional space onto a linear m-dimensional subspace ($m < n$) can result in a large error in the representation, particularly in a high nonlinear feature space such as image visual features. Therefore, under such circumstances, an (NLDR) can yield a better performance compared to an LDR. The basis of NLDR is the standard non-linear regression analysis. The learning phase of a neural network can be described mathematically as the problem of discrete, nonlinear optimization, In recent years, the neural network approach for NLDR has been widely studied. The advantage of using neural network for NLDR is that it can learn directly from input sample data to get the expected solutions. It can also extract the true model of feature data from a limited set of limited training samples. In this chapter, we review some of basic neural network models that can be used for feature dimension reduction. These basic neural network models can be used to construct complex systems.

4.2 Overview of Neural Network

4.2.1 Basic Structure of a Neural Network

Artificial neural network models have been studied for many years in the hope of achieving human-like performances in the fields of speech and image recognition where many hypotheses are pursued in parallel, high computation rates are required, and the current systems are far from equaling human performance. A neural network model is a biologically inspired computational structure composed of many simple, highly interconnected computational elements or nodes as shown in Figure 4.2.1.

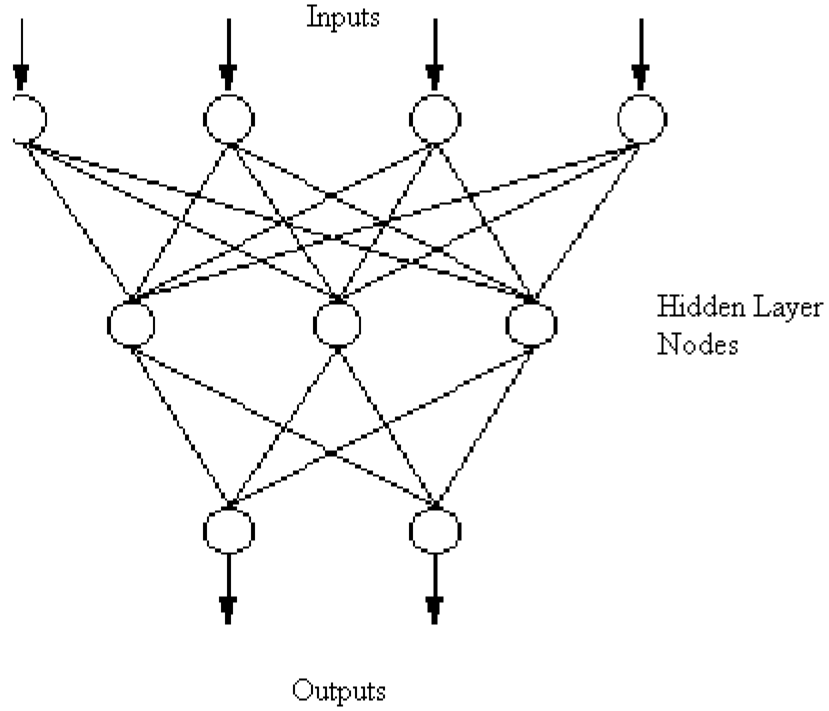


Figure 4.2.1 A typical neural network model

These computational elements or neurons are connected via weights and typically execute in parallel and exchange information much like the neurons and synapses within the brain. Each neuron, as shown in Figure 4.2.2, typically receives signals from several computational elements, processes this information, and passes a signal on to several more neurons in a manner analogous to biological neurons.

Each neuron used in a neural network is typically nonlinear. The simple neuron sums N weighted inputs and passes the result through a nonlinear function. The three common types of nonlinear are hard limiters, threshold logic and sigmoidal function.

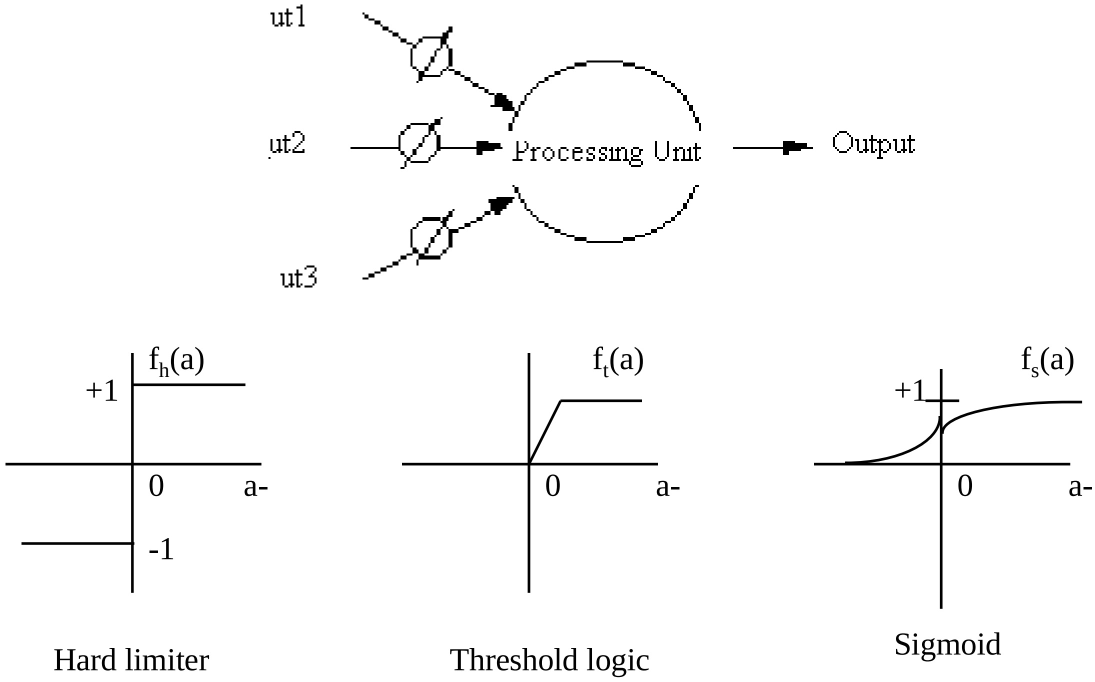


Figure 4.2.2 Network neurons and three representative nonlinearities

Neural network models are specified by the network's topological node characteristics, and the network training or learning rules. The network learning rule specifies an initial set of weights and indicates how the weights should be adapted during training. It has a great effect on the network performance. Many neural network architectures and network learning rules have been proposed and are still major topics of current neural network research.

Instead of executing instructions sequentially as in a traditional computer, neural networks explore many competing hypotheses simultaneously using parallel nets composed of many computational elements connected by weights. The main feature of neural network techniques is "learning by example" which set them apart from conventional methods. Network Learning is accomplished by presenting input and output pairs and adjusting the weights, or strength between the connections, of the network in order to minimize errors over a set of example inputs and outputs. The data sets which consist of the input and output pairs during learning are called network training data sets. Other sets of data containing input and output pairs not used during training are referred to as testing sets. The testing data set is used to test the network training performance, which is usually measured as the root-mean-square (RMS) error. The RMS indicates the ability of the network to generalize from the training inputs to new, previously unseen inputs.

4.2.2 Neural Network Training

There are many learning methods of neural networks available. They can be divided into supervised and unsupervised learning. Supervised learning is the most common training method where there is a "teacher" who in the learning phase "tells" the net how well it performs ("reinforcement learning") or what the correct behavior would have been ("fully supervised learning"). In supervised learning, the training set completely specifies all input examples as well as the outputs which are desired when those inputs are presented to the neural network. The neural network can be trained by presenting the training samples to the network one at a time. The network’s weights are updated in such a way that the measures of the error in the network result are minimal. A pass through the whole training set sample, alone with an updating of the network’s weights, is sometimes called an epoch. The number of epoch required to train the network until the network’s performances satisfactory is used to measure the convergence speed of network.

In unsupervised learning, the net is autonomous. It just looks at the data it is presented with, finds out some of the properties of the data set and learns to reflect these properties in its output. What exactly these properties are that the network can learn to recognize, depends on the particular network model and learning method. Usually, the net learns some compressed representation of the data. Unsupervised learning allegedly involves no target values. In fact, for most varieties of unsupervised learning, the targets are the same as the inputs. Although supervised learning is the most popular method used by many network models, in some circumstances, unsupervised learning can be especially valuable in exploratory work. For example, distinct classes are assumed to exist in a collection of samples although it is not sure what those classes are in.

Unsupervised learning is used in a wide variety of fields under a wide variety of names such as cluster analysis, Hebbian learning [Hertz, *et al.*, 1991] in which multiple output units are often active together, Kohonen's self-organizing (feature) map SOM [Kohonen, 1995] and competitive learning in which only one output unit, or only one per group, is active at a time. Cluster analysis appears as Adaptive Vector Quantization (AVQ) [Kosko, 1992] in neural network literature. Each of the hidden units in AVQ corresponds to a cluster center. The error function is the sum of squared distances between each training case and the nearest center. If the error function is the same error function used in k-means clustering [MacQueen, 1967; Anderberg, 1973; Balakrishnan, *et al.*, 1994], the neural network performs the k-means clustering function.

Hebbian learning is unsupervised learning. It has the same function as an auto-associative network with a linear layer hidden by minimizing the same error function. The network is trained by least square means (LSM) and is most suitable for dimensionality reduction. The sum of the squared distances between each training case and a linear subspace of the input space (with distances measured perpendicularly) is called an error function which is minimized by the leading principal components exactly [Joliffe 1986; Jackson 1991]. In fact, Hebbian learning can produce principal components [Hertz, *et al*. 1991; Karhunen 1994; Deco and Obradovic 1996]. The other form of unsupervised learning is SOM [Kohonen 1995] which combines competitive learning with dimensionality reduction by smoothing the clusters with respect to an *a priori* grid [Kohonen 1995; Mulier & Cherkassky 1995].

There is another training method called hybrid-training method. It is unsupervised learning in that the exact outputs desired are not specified. However, it is supervised learning in that it is told whether its response was good or bad when the network responds to a sample in the training set. Hidden weights of hybrid methods are first obtained by unsupervised learning, after which the output weights are obtained by supervised learning. The hybrid training approach can thus be much faster than the nonlinear optimization that would be required for supervised training of all of the weights in the network.

Hybrid training will usually require more hidden units than supervised training since supervised training optimizes the locations of the centers, while hybrid training does not. Supervised training will provide a better approximation to the function to be learned for a given number of hidden units. Thus, the better fit provided by supervised training will offer fewer hidden units for a given accuracy of approximation than with hybrid training. If the hidden output weights are learned by linear least-squares, the fact that hybrid training requires more hidden units implies that hybrid training will also require more training cases for the same accuracy of generalization (Tarassenko & Roberts, 1994).

In summary, unsupervised learning is faster than the supervised learning but it has less accuracy. Hybrid learning is a combination of two learning methods to achieve accuracy as well as fast learning results.

4.2.3 Neural Network Capabilities

Neural networks (NN) have great potential for many applications. In practice, NNs are especially useful for classification and function approximation or mapping problems which are tolerant of some imprecision and have lots of training data available, but to which hard and fast rules (such as those that might be used in an expert system) cannot easily be applied. Almost any mapping between vector spaces can be approximated to arbitrary precision by feedforward NNs (which are the type most often used in impractical applications) if there are enough data and computing resources. In recent years, neural network has been applied to feature extraction or feature dimension reduction. The architecture of the neural network for NLDR can be classified into three categories: auto-associative, self-organizing and standard multi-layer neural networks.

4.3 Auto-Associative Neural Networks for Dimension Reduction

Auto-associative networks are special multi-layer neural networks which uses identical unary input and output patterns. These networks consist of an input layer that has *N* units to receive raw data vectors; a hidden layer with *M* units which is assumed to be much smaller than the dimension of the input vector ($M < < N$); and an *N* output units layer. The network weights and biases are trained by error backpropagation with the target outputs equal to the inputs. This is often refered to as an *N*-*M*-*N* encoder. Figure 4.3.1 shows the architecture of a three-layer feedforward auto-associative network. The network is trained on a set of n-dimensional real-valued vector x_n such that each x_n is mapped to itself at the output layer in an auto-associative mode. Thus the network is trained to act as an encoder of real-valued patterns. There are exactly *N* members of the training set ($p = N$), each having one input with the corresponding active target, and the rest inactive [Hertz, *et al.*, 1991].

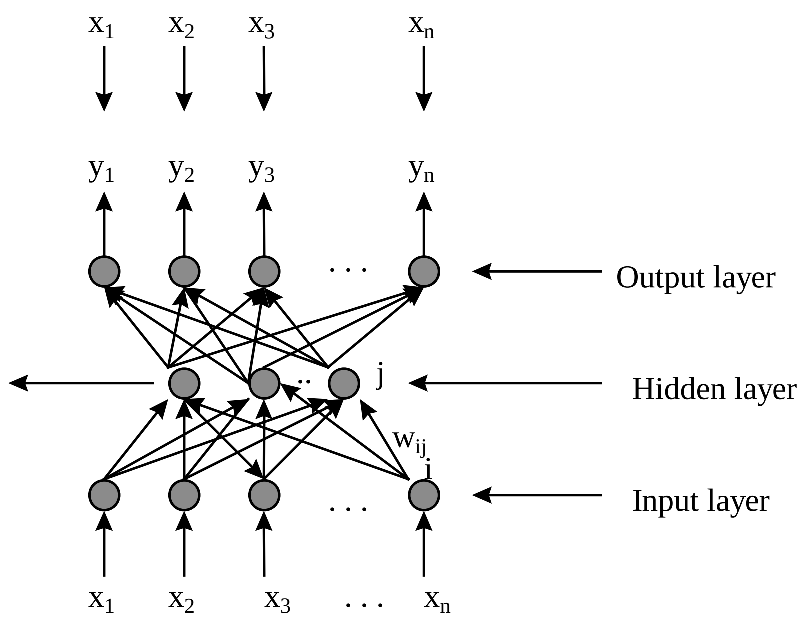


Figure 4.3.1 A three-layer feedforward autoassociative network

Several researchers have extended this network for dimension reduction [Garrison *et al.,* 1991; Golomb *et al.,* 1991]. The concept of an auto-associative neural network for dimension reduction used a lower dimension to explain maximum information. Since the hidden layer which has fewer nodes than the input layer , after the network training is completed, the hidden units of network can be used as the representation of the lower dimensions of raw data sources. In these networks, if the units of the networks have a linear activation, it is called a linear auto-associative neural network. It is very interesting that the linear auto-associative neural network actually performs PCA transformations [Baldi & Hornik, 1989] and the dimensions of its hidden layer equal the number of principal components. For nonlinear dimension reduction, a nonlinear activation is required. However, even with nonlinear activation, a three-layer auto-associative network cannot achieve compression superior to a PCA [Bourlan & Kamp, 1988].

Some researchers have tried to use a five-layer auto-associative network to improve the performance of a three-layer auto-associative network for nonlinear dimension reduction [Usui *et al.,* 1991; Oja, 1991, Kramer, 1991; Namphol *et al. 19*91]. The architecture of a five-layer auto-associative network is shown in Figure 4.3.2. The network has five layers. They are: input, mapping, bottleneck, de-mapping and output layers. The nodes of the mapping layer and the de-mapping layer have nonlinear activation, which the nodes of the other three layers have linear activation. A five-layer auto-associative network can also be seen as the combination of two three-layer networks. Thus the network can be trained separately, so the training is straightforward.

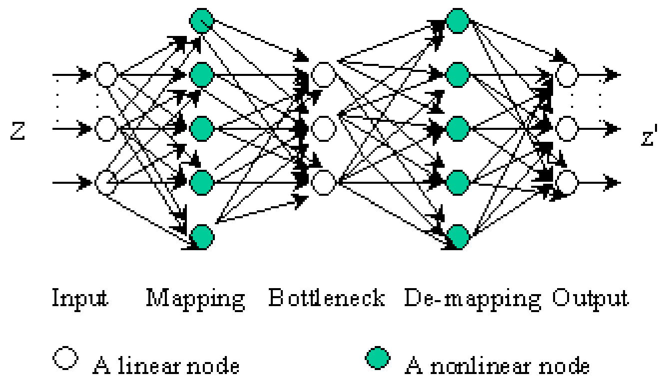


Figure 4.3.2 The architecture of a five-layer auto-associative network

The objective activation function used to train the network is:

$$E = \sum_{j=1}^{n}\sum_{i=1}^{m}(Z_i - Z_i^{'})_j^2 \qquad (4.\ 3.\ 1)$$

where Z_i is an element in the data set Z and $Z_i^{'}$ is an output of the network. To train the network, the weights appearing in the network are optimized so that the reconstructed outputs Z' match the input Z as closely as possible. Training is complete when E, the sum of squared errors given in Equation 4.3.1, is minimized. The actual training can be approached by any appropriate learning algorithm such as backpropagation [Rumelhard *et al.*, 1986].

After network training is completed, since the nodes of bottleneck layer are less than the input and output layers nodes, the network is forced to develop a compact representation of the input data. The bottleneck layer is refered to as the representation layer. The nodes of the representation layer construct coordinates on a generally curved, submanifold of the input space. Since the representation layer maps these coordinate values back into the input space, the network can cover the entire domain of the input space.

It has been claimed that the performance of this network can yield better results for nonlinear dimension reductions than three-layer auto-association networks. Applications of this nonlinear auto-associative network include noise reduction, replacement of missing sensor values, and error detection and correction. However, the limitation of this auto-association network is that the networks can only be trained through connections between units whose target activation can be specified from outside the network. Moreover, it is be very difficult to train the network since the network has five layers. Finally, it can be difficult to determine the number of nodes in mapping, de-mapping, bottleneck layers.

4.4 Self-Organizing Neural Networks for Dimension Reduction

A self-organizing neural network is an unsupervised learning neural network that organizes data to be presented during the learning phase, where the result of mapping is not required. The self-organizing neural network adapts its internal knowledge to the task following the training rules. The aim of this kind of network is to cluster or categorize input data. Similar inputs are classified as being in the same category.

A typical self-organizing neural network is the Kohonen map introduced by Kohonen [1984, 1990]. The Kohonen map is a two-layer network. The input patterns are organized in a topological structure represented by its units, where the relations between different patterns are preserved. In this map, the units of the first layer receive raw data inputs and are connected to a second *n*-dimensional competitive layer, where the competitive learning system is used. In the most general case, each unit in the competitive layer receives input from each unit in the layer immediately below it and projects to each unit in the layer immediately above it. Within a competitive layer, the units are broken into a set of inhibitory clusters in which all elements within a cluster inhibit all other elements in the cluster. Thus the elements within a cluster at one level compete with one another to respond to the pattern appearing on the layer below. The more strongly any particular unit responds to an incoming stimulus, the more it reduces the response of the other members of its cluster. Figure 4.4.1 shows the architecture of a two-dimensional Kohonen map.

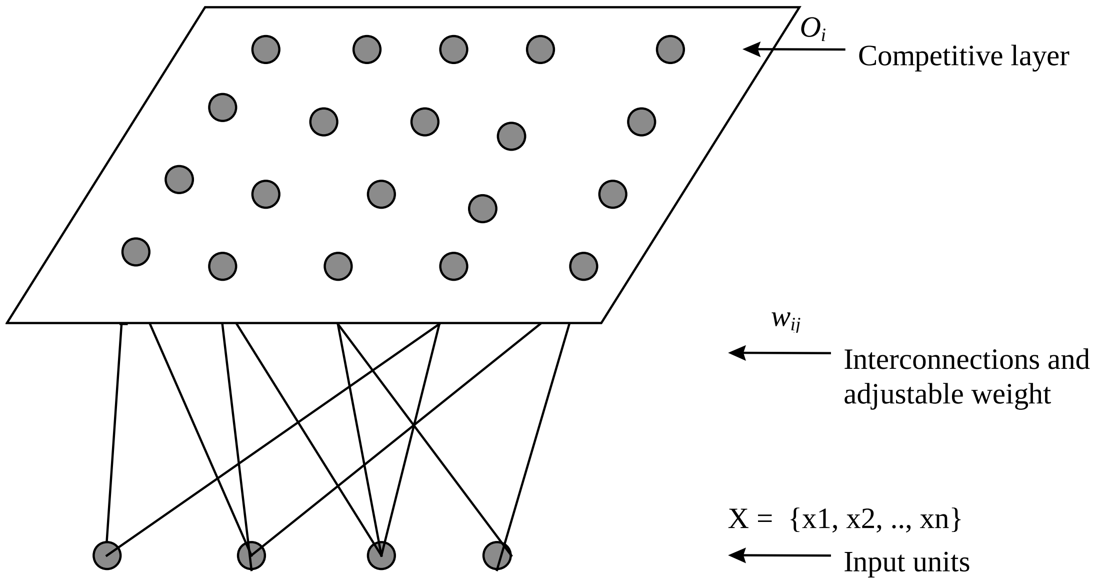


Figure 4.4.1 The architecture of Kohonen map

The interconnections' weights are adjusted during training and only one single unit in the competitive layer is active at a time. To train the network, an *n*-dimensional continuous-valued input data vector X = $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ is presented to the input layer sequentially in time without specifying the desired output. The output units O_i are arranged in an array, and are fully connected via weights w_{ij} to the input units. After enough input vectors have been presented, weights will specify clusters or vector centers. The clusters sample the input space so that the point density function of the vector center tends to approximate the probability density function of the input vectors [Kohonen, 1984]. Since the weights are organized in such a way that topologically close nodes are sensitive to input that are physically similar, output nodes will thus be ordered in a natural manner.

The Kohonen map can be trained by Kohonens' learning algorithm [Hertz, 1991] which take a computational shortcut to achieve the effect accomplished by the 'Mexican hat' lateral interactions. The weight update rule is modified to involve neighborhood relations in the output array. A competitive learning rule is used which will choose the winner i^* as the output units with the weight vector closest to the current input x, such as:

$$|\boldsymbol{w}_{i^*} - X| \leq |\boldsymbol{w}_i - X| \quad \text{for all } i. \tag{4.4.1}$$

For all i and j, the learning rule can be derived by:

$$\Delta w_{ij} = \eta\ \varphi(i, i^*)(x_i - w_{ij}) \tag{4.4.2}$$

The neighborhood function $\varphi\ (i,i^*)$ is 1 for $i = i^*$ and falls off with distance $|\ r_i - r_{i^*}|$ between units i and i^* in the output layer. Thus units close to the winner, as well as the winner i^* itself can have their weight change gradually, while those further away ($\varphi\ (i, i^*)$ is small) experience little effect.

To construct a practical algorithm, a neighborhood function is required to be defined around each node which is shown in Figure 4.4.2. In Figure 4.4.2, N_i (t) is the set of nodes considered to be in the neighborhood of node i at time t. The neighborhood starts large and slowly decreases in size over time ($t_0 < t_1 < t_2$). After the neighborhood function is defined, the weights between input and output nodes are initially set to small random values and the input is presented. The distance D_j between the input and output and all nodes is computed as follow:

$$D_j = \sum_{i=0}^{N-1} (x_i(t) - w_{ij}(t))^2 \tag{4.3.3}$$

where $x_i(t)$ is the input to node i at time t and $w_{ij}(t)$ is the weight from input node i to output node j at time t. If the weight vectors are normalized to a constant then the node with the minimum Euclidean distance can be found by using the net of Figure 4.3.1 to form the dot product of the input and the weights. The selection which is required for minimum D_j becomes the problem of finding the node with a maximum value. Once this node is found, weights to it and to other nodes in its neighborhood are modified according Equation 4.3.3 to make these nodes more responsive to current input. This process is repeated for further inputs until the weights eventually converge and are fixed.

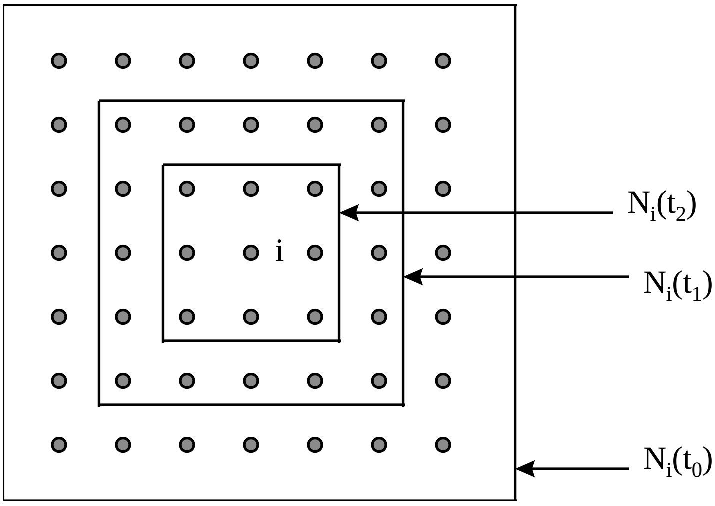


Figure 4.4.2 Topological neighborhoods at different times as the maps are formed

A Kohonen map is also called a self-organizing feature map where high-dimensional data are mapped to two dimensions using a neural architecture. In general, the Kohonen map uses the classical iterative optimization techniques, such as k-means clustering [Duda & Hart, 1973]. Since the units in the output layer are ordered topologically, *i.e.*, neighboring units react to data vectors similar to the input space, the mapping can be seen as the reduction of an *n*-dimensional space into *m*-dimensional spaces ($m << n$), maintaining the maximum topology similarity of data in *n* and *m* dimensional space. Generally, the output layer is either one or two dimensions.

The disadvantage of a Kohonen map is that all data sets, regardless of their intrinsic dimensionality, are presented in two dimensions. Thus the user has no control over the loss of information incurred in the mapping process. Some researchers have been trying to extend the Kohonen map to any dimension. For example, Markus and Seibert [1993] extended the standard Kohonen map into a *3*-dimensional space in order to visualize a multidimensional data set in a three dimension colour space *e.g.,* each multidimensional data point corresponds to a colour triplet (*R, G, B*). Because the network arranges the units topologically during training, neighboring units with similar colour reflect the neighboring data in the multidimensional space. Thus the similarity of multidimensional data can be perceived by the human visual system.

Auto-association neural networks and self-organizing neural networks are comparable in that they both amount to searching for a lower dimensional interpretation of high-dimensional data using similar cost functions with a landscape smoothing parameter. However, they can fall into incorrect local minimal and may fail when the constrained surface doubles back on itself sharply. Moreover, these methods are based on unsupervised learning rule. The knowledge of outside networks such as human visual perceptions in image databases can not be integrated into the network's weights.

4.5 Multi-Layer Perceptron Neural Networks for Dimension Reduction

Multi-layer perceptron neural networks are feed-forward nets with one or more layers of nodes between the input and output nodes. These additional layers contain hidden units or nodes in which nonlinearities are used. If the nodes were linear activation function, then a single-layer net with an appropriately chosen weight could exactly duplicate those calculations performed by any multi-layer perceptron neural network. These hidden units or nodes are not directly connected to both the input and output nodes. Multi-layer perceptron neural networks can be trained by any supervised learning algorithm.

The capabilities of multi-layer perceptron neural networks stem from the nonlinearities used within nodes. A three-layer perceptron neural network can form arbitrarily complex decision regions and can separate the meshed classes. It can form regions as complex as those formed using mixture distributions and nearest-neighbor classifier [Duda & Hart, 1973]. It has been proved that no more than three layers are required in perceptron-like feed-forward nets since a three-layer net can generate arbitrarily complex decision regions. Thus a three-layer perceptron-like feed-forward nets classifier with multiple output nodes and nonlinearities used within nodes can be used for NLDR.

4.5.1 The Architecture

A three-layer feed-forward net for NLRD is given in Figure 4.5.1.1. The network consists of a number of units which are connected by weighted links. The units are organized in three layers, namely: input, hidden and output. The input layer receives raw data vectors and passes it through interconnected weights to the units of the hidden layer. The features presented to the input layer can be single or a combination of image feature vectors for training the network. The hidden layer is configured with fewer units than the input layer. In general, the dimensions of the lower feature space determine the number of hidden units. The number of units in the output layer equals to the number of training classes and the number of units in input layer corresponds to the dimensions of the raw data vector. Each layer can be fully interconnected with associated weights.

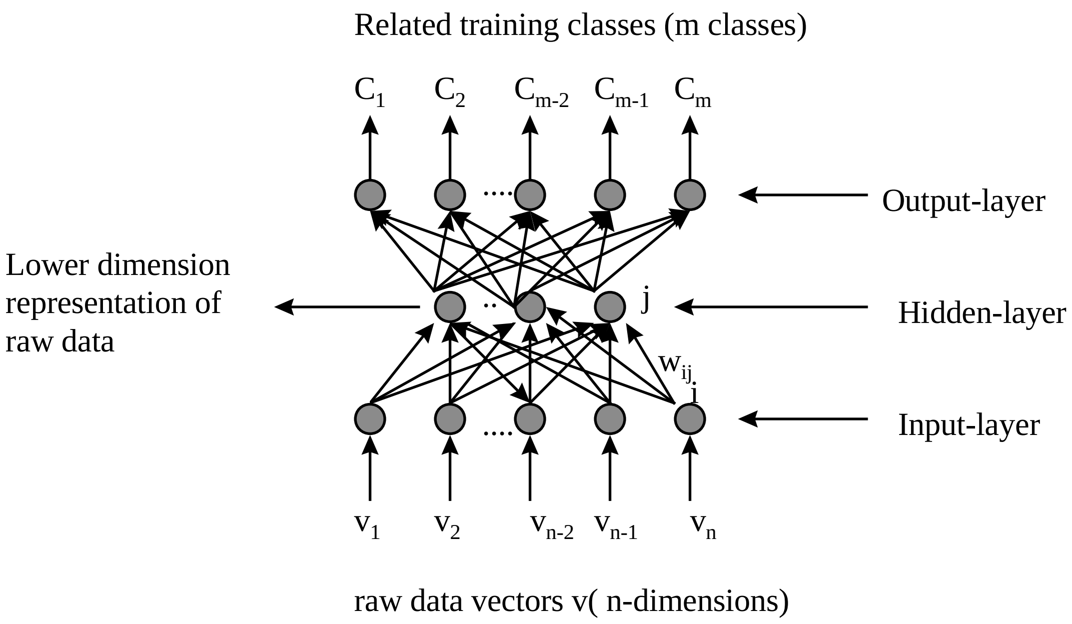


Figure 4.5.1.1 Topology of a three-layer network classifier for NLDR.

Raw data vectors are presented to the input layer then propagated forward through the hidden layer and yield an output activation in the output layer. The output activation correspondent to one of the training classes.

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* can be given:

$$\text{net}_i = \sum_{j \in P(i)} s_j w_{ij} - \theta_i \qquad (4.5.1.1)$$

where w_{ij} is the interconnected weights from unit j to unit i, and θ_i is the unit's bias value. θ_i is often replaced by a weight to a 'bias unit' with a constant output of one because of the homogeneous representation. Passing the net$_i$ through a non-linear activation function, the unit i activation s_i can be obtained. Usually, the sigmoid logistic function is used such that:

$$s_i = f_{\log}(net_i) = \frac{1}{1+e^{-net_i}} \qquad (4.5.1.2)$$

Thus the derivative can be computed easily as the following:

$$\frac{\nabla s_i}{\nabla net_i} = f_{\log}^{'}(net_i) = s_i * (1 - s_i) \qquad (4.5.1.3)$$

4.5.2 Supervised Learning Rules

The advantage of supervised learning is that the network's weights can be adjusted such that the network performs a desired mapping of input to output activation. The mapping is given by presenting a set of predefined training patterns. These predefined training patterns can be selected based on human visual perceptions in the dimension reduction of image features. Each pattern pair p in training set $\Re$ consists of an input vector I_p and desired output vector O_p. The goal of the supervised learning is to adjust the network's weights such that the following error function E is minimized over the training set $\Re$:

$$\text{E} = \sum_{p \in \Re} \sum_{n} (O_n - I_n)^2 \qquad (4.5.2.1)$$

Where p denotes all the pattern pair in the training set and n is the number of units in output laye s_n is the output activation of network.

The rule to update the weights of the network can be defined as follows:

$$\Delta w(t) = \eta\, d_m(t) \tag{4.5.2.2}$$

$$w(t+1) = w(t) + \Delta w(t) \tag{4.5.2.3}$$

where η is the learning parameter controlling the learning rate, and $d_m(t)$ is the direction that the weight needs to be adjusted in order to minimize the cost function E. The common way to determine the search direction $d_m(t)$ is to use a first order derivative namely the gradient ($\Delta E = (\partial E / \partial w)$). There are many learning algorithms for determining ΔE. The backprop algorithm is the most important learning algorithm in multi-layer feed-forward neural networks. It performs successive computations of ΔE by propagating the error back from the output layer towards the input layer. In the next section, a brief overview of the backprop algorithm is given.

4.5.3 The Backprop Algorithm

The basic concept of the backprop algorithm is to first compute the partial derivatives $\partial E / \partial w_{ij}$ and then compute the result weight update by using the gradient descent technique. For each of the weights in the network, the partial derivatives can be computed repeatedly by the following rule:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial s_i}\frac{\partial s_i}{w_{ij}} \tag{4.5.3.1}$$

where

$$\frac{\partial s_i}{w_{ij}} = \frac{\partial s_i}{\partial net_i}\frac{\partial net_i}{\partial w_{ij}} = f_{\log}'(net_i)s_j \tag{4.5.3.2}$$

There are two cases for computing the $\partial E / \partial w_{ij}$ which is the influence of the output s_i of unit *i* on the global error function E.

1. If *i* is an output unit, the $\partial E / \partial s_i$ can be simply obtained as:

$$\frac{\partial E}{\partial s} = \frac{1}{2}\frac{\partial (O_i - I_i)^2}{\partial s} = -(O_i - I_I) \qquad (4.5.3.3)$$

2. If *i* is not an output unit, then a recursive rule is required to compute the $\partial E / \partial s_i$, which is given as follows:

$$\begin{aligned}
\frac{\partial E}{\partial s_i} &= \sum_{l \in succ(i)} \frac{\partial E}{\partial s_l}\frac{\partial s_l}{\partial s_i} \\
&= \sum_{l \in succ(i)} \frac{\partial E}{\partial s_l}\frac{\partial s_l}{\partial net_i}\frac{\partial net_i}{\partial s_i} \\
&= \sum_{l \in succ(i)} \frac{\partial E}{\partial s_l} f_{\log}'(net_i) w_{li} \qquad (4.5.3.4)
\end{aligned}$$

where *succ (i)* is the set of all units *l* in successive layers (closer to the output layer) to which unit *i* has a non-zero weight connection w_{li}. Thus the gradient information is successively fed back from the output layer to the input layer. This is called the back-propagation algorithm or backprop algorithm.

After the partial derivative $\partial E / \partial w_{ij}$ is computed, a gradient descent technique is applied so that the weight update is a scaled step in the opposite direction of the gradient. In other words, the weight update is equal to the negative derivation that is multiplied by a constant value. That constant value is called learning rate ε. For a single weight, the weight update can be calculated as follows:

$$\Delta w_{ij}(t) = -\varepsilon * \frac{\partial E}{\partial w_{ij}}(t) \qquad (4.5.3.5)$$

It is sometimes difficult to decide the value ε because a small ε can result in a slow convergence and a large ε will led to oscillations. Moreover the learning may be unstable. For example, a shallow error-function can result in a small weight step, while a sharp error-function may result in a large weight step. To overcome these limitations, one solution is to add a momentum term to Equation (4.5.3.5) such that:

$$\Delta w_{ij}(t) = -\varepsilon * \frac{\partial E}{\partial w_{ij}}(t) + \delta * \Delta w_{ij}(t-1) \qquad (4.5.3.6)$$

Thus the momentum scales the influence of the previous weight-step upon the current one.

Weight-update can either be performed after each presentation of a pattern pair and computation of the respective gradient which is refereed to as 'learning by pattern' or performed after the presentation of whole training pattern set which is called 'learning by epoch'. The method of learning by epoch is the most commonly used because the summed gradient information for the whole training set contains more reliable information for the shape of an entire error-function.

The major steps involved in backprop algorithm are:

1. Initialize the weights to small random values.
2. Set the learning rate ε to a proper positive value.
3. Select an input training pattern I_k and propagate it through the network.
4. Employ Equation (4.5.3.3) to compute the gradient of output units and Equation (4.5.3.4) to compute the gradient of other units.
5. Use Equation (4.5.3.5) to update all connection weights according to $w_{ij}^{new} = w_{ij}^{old} + \Delta w_{ij}$.
6. Test for convergence. This is done by checking some pre-selected function of the output errors to see if its magnitude is less than a threshold. If the condition is satisfied, stop; otherwise, go back to step 2 and repeat the pattern.

In general, learning with the backprop algorithm is slow [Huang & Lippmann, 1988] due to the characteristics of error surface. Error surface is characterized by numerous flat and steep regions. Moreover, the error surface has many troughs that are flat in the direction of minimum search. Any of these features are particularly critical in classification problems, especially when the size of the training set is small [Hush *et al*., 1991]. Many enhancements of and variations to the backprop algorithm have been proposed in recent years. The goal of these heuristic modification algorithms is to increase the speed of convergence, avoid local minimum, or improve in the network's ability to generalize.

The quickprop algorithm [Fahlmam, 1988] is a modification to the backprop algorithm. This algorithm has the same procedure as the backprop algorithm except for each weight a copy of $\partial E / \partial w_{ij}$ is kept. The weights update then can be obtained by the following equation:

$$\nabla \mathrm{w_{ij}}\,(\mathrm{t}) = \frac{\frac{\partial E}{\partial w_{ij}}(t)}{\frac{\partial E}{\partial w_{ij}}(t-1) - \frac{\partial E}{\partial w_{ij}}(t)} \Delta w_{ij}(t-1) \qquad (4.5.3.6)$$

where $\frac{\partial E}{\partial w_{ij}}(t-1)$ and $\frac{\partial E}{\partial w_{ij}}(t)$ are previous and the current values of $\partial E / \partial w_{ij}$. The computation of (4.4.3.6) is easy and straightforward.

The use of error gradient information at two consecutive steps in Equation (4.5.3.6) to improve convergence speed can be seen as being based on approximations of second-order search methods such as Newton's method. It is important to note that this second-order modification to the backprop greatly improves the learning time compared with standard backprop algorithms. Indeed, the quickprop algorithm is the most frequently used adaptive learning paradigm so far [Riedmiller, 1994].

4.6 The Application of Multi-layer Neural Networks for Dimension Reduction

A three-layer perceptron neural network with a quickprop learning algorithm is used to perform the dimension reduction of image features for image databases. The network actually performs as an image classifier. In order to train the classifier, a training data set which includes many training patterns is required from the image database. A training pattern pair (I_v, ID(i)) in the training data set consists of the image feature vector and its correspondent class ID. The image feature in the training pair can contain either one feature vector or multi-feature vectors. The class ID of an image can be defined by a domain expert based on human visual perceptions. It means that a set of images that is perceived similarly by human beings should have the same class ID. The format of a class ID is a string of binary numbers in which only one bit has value '1' in each class ID, and the rest of the bit is 'o'. The bit having the value '1' represents the class to which the image belongs. Therefore the number of bits used in class ID is equal to the number of the image classes.

The procedure of network training consists of repeated presentations of input and desired output pairs to the network. At each training trial, the individual feature vectors can be used first to train the network and then to include the other feature vectors. Alternately, a combination of a multi-feature feature vector can be used at each training trial. The network performs classification tasks. During the network training, network weights accumulate the knowledge from the training data sets and approximates an inlay mapping on the set of input vectors. The accumulated knowledge is, in this case, the weighting of different image features based on human visual perceptions.

After the network is trained successfully, since the hidden layer has fewer units than the inputs layer, the network is forced to develop a compact representation of input data in the hidden layer. Thus the dimension reduction of features can be achieved by taking the units in the hidden layer as the representation of the input data set in the lower dimension space. For every image, passing it through the network and taking the value of the hidden units from the network, a new compressed image set is formed for the original image set. This new compressed image set keeps the maximum information about the weighting among image features. Therefore the question about how to find two images which are similar based on the combination of multi-visual features turns into the problem of fining the closest distance between two images in the compressed set. Thus any of the mathematical distance functions can be used to measure the similarity of images based on the combination of multi-visual features.

Below is an example which illustrates how a three-layer perceptron neural network can be used for the dimension reduction of an image feature. The data for this example consists of 30 texture feature variables, which are derived from nine images from Appendix 1. These nine images can be grouped into three classes based on human perceptions of the image similarity and are used as the training set. Table 4.6.1 shows the complete training set. V_i denotes the i*th* image texture vector (30-dimensions). Class ID has three bits which represent image classes. (For example, bit 1- denotes class 1, bit 2 --denoted class 2 and bit 3 – denotes class 3)

Table 4.6.1 Training set

	Input	output	Classes
Image1	v1	001	1
image2	v2	001	1
image3	v3	001	1
image4	v4	010	2
image5	v5	010	2
image6	v6	010	2
image7	v7	100	3
image8	v8	100	3
image9	v9	100	3

The network has 30 input units corresponding to the 30 dimensions of texture; the number of output units is three which are equal to the number of the image classes. The number of hidden units depends on how many dimensions of the lower feature space are required. We only select two hidden units so that the results of dimension reduction can be directly visualized in a two-dimension space. If the distance of two images is close in low dimension space, they should be close in high dimension space if the same dimension reduction methods are used. In practical applications, the number of the hidden nodes can be chosen in the range of 4-5 so that more information can be maintained.

The weights of the networks are randomly initialized within the range [0,1] and the activation of the network is a nonlinear asymsigmoid function. The quickprop algorithm is used as the network learning algorithm. There are two important parameters in the quickprop algorithm *i.e.* a learning rate ε for the gradient descent, and a second parameter v which limits the step-size. We select $\varepsilon=0.5$ and $v=1.75$. Table 4.6.2 shows the result of nine images after dimension reduction.

Table 4.6.2 2-D representations of nine images

	x	Y
Image1	0.573975	0.667465
image2	0.537993	0.723921
image3	0.597284	0.761829
image4	0.142230	0.854514
image5	0.218078	0.961721
image6	0.054934	0.750447
image7	0.266506	0.120612
image8	0.193800	0.028562
image9	0.270177	0.160220

Figure 4.6.1 shows a two-dimension plot of this data set. It can be seen from Figure 4.6.1 that three classes of image are separated very well in the two-dimension space. The distance of any two points in one cluster is less than the distance of any other two clusters. Thus the measurement of the image similarity can become the measurement of distance among the points in feature space. Image retrieval based on this method can overcome the problem of false dismissal (*i.e.*, all qualifying images may not be retrieved) and false alarms (*i.e.*, images which are qualified may be returned).

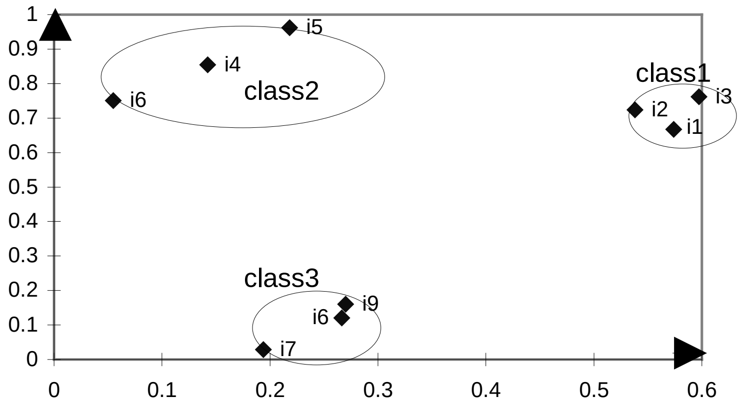


Figure 4.6.1 Plot of *9* image features with respect to their 2-D representations

4.7 Concluding Remarks

In recent year, the neural network approach for NLDR has attracted many researchers' attention. An overview of basic concepts of neural networks was presented in this chapter. Three basic neural network models for NLDR were also discussed. The advantage of using a supervised learning network for NLDR in image retrieval is that it can learn the knowledge of human visual perception directly from input sample data to get the expected solution.

An example of using a three-layer perception neural network with supervised learning for reducing the dimensions of image features was also given in this chapter. In this example, a neural network is used as a pattern classifier. The result in Figure 4.6.1 indicates that the measurement of similarity among images can be transformed to a measurement of mathematical similarity through a three-layer perceptron neural network. The mapping of the neural network is sufficient for distinguishing image similarity in a lower feature space. However, the network training could be slow in particular, with increasing input units. Speeding up the neural works training is a major research effort in neural network area. In the next chapter, we introduce a hybrid diminution reduction method which aim to improve the neural network training performance.

Chapter 5

The Hybrid Dimension Reduction

In this chapter, a new fast robust method, the hybrid dimension reduction is proposed for NLDR of image feature. This hybrid method combines a nonlinear neural network with a linear PCA to achieve a fast, accurate NLDR. This method takes advantage of the nonlinearity property of a neural network with supervised learning to achieve accurate nonlinear dimension reductions and utilizes a property of the PCA to reduce the dimension of the network input to improve the training speed. In neural network training, the degree of contribution of each component of feature to the classification is not known beforehand, The PCA is an ideal candidate to be used to gain the contribution information of each component in the input vector.

5.1 Introduction

The principal component analysis discussed in Chapter 3 is a well known classical statistical technique used for linear data analysis and dimension reduction. Since the PCA is based on second moments it is unable to detect and eliminate higher-order nonlinear redundancies in the data space. In fact, its projection models the original data set as an m - dimensional Gaussian signal plus n - m degrees of freedom for noise. The elements of the dimension reduction vectors construct global coordinates on the m - dimensional hyperplane which is spanned by the principal correlation eigenvectors. After the PCA linear transformation, the original n - dimensional data vectors lie near this hyperplane in a least mean square error.

This linear subspace model has some limitations that diminish its applications:

1. A standard PCA is able to realize only linear input-output mapping.

2. The principal components are defined solely by the data correlation (covariances). These second-order statistics only characterize Gaussian data and stationary, linear processing operations.
3. PCA can not separate independent subsignals from their linear mixture.
4. The outputs of standard PCA are usually at most mutually uncorrelated but not independent. This is not desirable in many applications.

These limitations of PCA make it unsuitable for high nonlinear data sets, such as image feature vectors which are not Gaussian distributed and nonlinear data samples. Using a linear dimension reduction of PCA for a nonlinear dimension reduction of image features can result in a larger error in the representation. The characteristics of the similarity of images no longer exist after PCA transformation. This problem of error mapping is highlighted in Chapter 3. However, the eigenvector required in a standard PCA can be computed efficiently using well-known numerical methods. Many researchers extend linear PCA to nonlinear PCA by using a nonlinear multi-layer neural network. The learning algorithm used in this nonlinear PCA is normally an unsupervised learning algorithm and the knowledge outside the network can not be integrated with the network weights. The performance of this nonlinear PCA for image compression can not achieve compression superior to the standard PCA [Bourlan & Kamp, 1988].

In order to reduce the dimension of image features, an NLRD is required. In general, the NLRD is refered to as a nonlinear regression problem which can be solved by a multi-layer neural network. There are many neural networks that can perform the NLRD tasks which are discussed in Chapter 4. The major argument for using a three-layer perceptron neural network for dimension reduction of image features is that the network contains nonlinearities which are more efficient for processing nonlinear image feature vectors. In addition, with a suitable supervised learning algorithm, the network can maintain a great deal of knowledge about image similarly reflected by human visual perceptions.

The major disadvantages of using a multi-layer neural network for dimension reduction of image features is that neural networks converge relatively slowly. In order to achieve good accuracy, it requires an excessive number of iterations with a large data set. The problem becomes worsen with increasing network inputs. There are two major methods to improve the network training speed: one is to reduce the input dimension of the network; the other is to adjust the learning parameters. A number of approaches to adjusting learning parameters to achieve a faster rate of convergence of neural networks have been proposed in recent years.

The traditional neural networks utilize all the dimensions of original input patterns for training and classification. However, a particular component of the input pattern does not necessarily contribute to classification and may even cause misclassification in some cases. Deletion of such a component from the training input can increase the performance of the network. However, it is difficult to find out which component of the training input vectors will have a bad effect on the recognition task beforehand. For example, in the CBIR system, the image features are extracted by image processing methods. From the point of view of image processing, it is important to extract the image features as much as possible which may in turn require a large number of models parameters. All these parameters may not be necessary for image similarity retrieval. Using all these parameters as the dimensions of the original network training input can lead to a long convergence time for the network and may cause the misclassification of image patterns.

This chapter presents the structure of a new hybrid method of dimension reduction and the related operations.

5.2 The Architecture of a Hybrid Dimension Reduction

The “black box” depiction of a hybrid dimension reduction is presented in Figure 5.2.1. The box contains knowledge of image similarity in the feature space. At the bottom of the black box, n input lines correspond to an n - dimensional feature vector. Out of the top of the box are k lines which represent the number of image classes. The m lines at the side of the box are feature vectors of lower dimensions. Each line of n, m represents a real number which is normalized among [-1, 1] and the value of the *k* lines is a binary number.

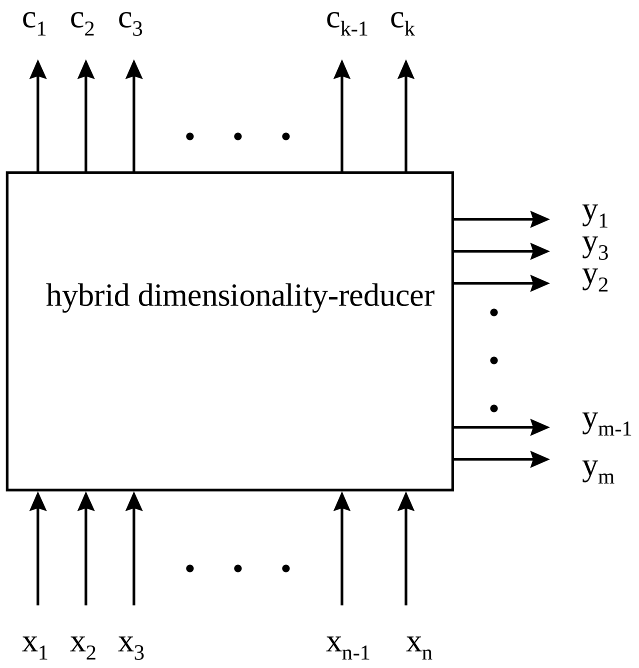


Figure 5.2.1 Black-box of hybrid dimensionality reducer

The dimensionality-reducer is first trained by a set of samples using a hybrid learning algorithm so that it can maintain the knowledge of the image similarity. Once the training is completed, the raw feature vectors are fed into the dimensionality-reducer and the corresponding lower dimensions of the new vectors are obtained from the side of the dimensionality-reducer in Fig 5.2.1.

The idea of this hybrid learning method is that the training data set for the network is first analyzed using the PCA. Then a new training data set called a PCA analyzed-data set is selected based on the direction of major eigenvectors of the correlation matrix of training data set in their order of significance. Thus the PCA-analyzed-data set contains lower dimensions than the original training set but keeps the maximum information about the training data. With the PCA-analyzed-data set, the computations of non-linear dimension reductions can be reduced dramatically and the network learning rate is also improved. This is due to the fact that most of the training iterations are conducted in the initial stages of the network learning algorithm, where the number of vector components used in the training is relatively small. This significant reduction in computation is traded off by the extra computation done by the PCA linear transformation of the training patterns. The computation of the PCA is straightforward by using well-know numerical methods. The time required for performing the PCA is rather small compared to the network training time. Figure 5.2.2 shows the architecture of this hybrid learning method for dimension reduction. The linear PCA appears on the bottom: the nonlinear neural network is on the top, and the representation of lower dimension vectors appears on the hidden layer.

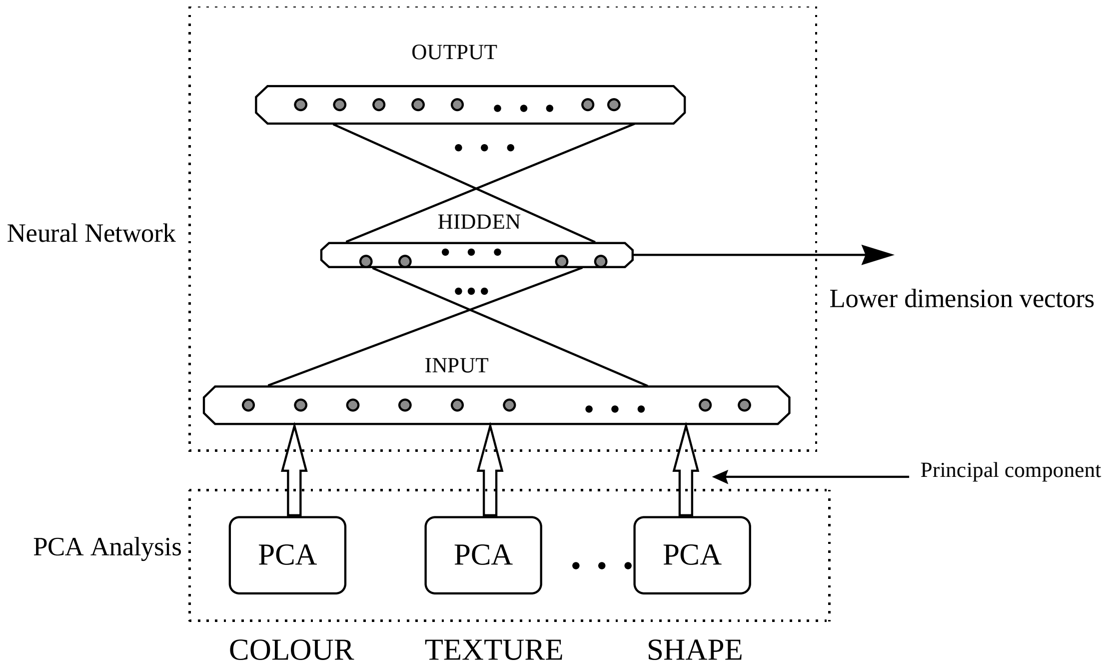


Figure 5.2.2 Hybrid scheme of dimension reduction.

At the bottom part of this structure, there are a number of boxes which contain the PCA analysis. Each feature of the images is analyzed by a single PCA box. The output of the PCA box are the corresponding principal component features which are used as input to the network. The SVD algorithm is used for computing the principal component of each image feature.

At the top layer of this structure is a neural network. This is the three-layer perceptron feedforward neural network already discussed in Chapter 4. This network serves as a standard pattern classifier. The architecture of the network is the same as in Figure 4.5.1.1. The input of the network is not the original data vector but the PCs which are derived from the PCA boxes underneath. Therefore, the number of units in the input-layer is decided by the number of PCs in the bottom layer. The network's outputs correspond to the image training classes. The activation function of this network is a nonlinear sigmoid logic function. The number of hidden units is determined by the dimensions of the lower feature space. The learning algorithm of this neural network is a supervised learning algorithm, Quickprop which has been discussed in Chapter 4.

The dimension reduction of image features consists of two steps. First, PCA analysis is performed on the visual features for all images. Based on the PCA, the first several PCs for each feature are selected to form intermediate representations of the original images. The number of PCs determines the number of the dimensions of the intermediate data representation. In order to recover the true lower dimension representation of an image from a nonlinear neural network, the number of PCs can not be too small. The PCs selected may depend on what they are used for. After the PCA, the nonlinear neural network reduction can be carried out on this intermediate representation of the image feature vectors. To train the network, a set of training samples is selected from this intermediate representation. The number of classes of the training set should be decided by the domain experts. The vector of the sample images presented to the neural network can be single feature vectors (texture only) or multi-feature vectors (texture and colour). After training, the hidden layer of the neural network can be seen as the final lower dimension representation of the images. The number of units in the hidden layer depends on the applications. In our case, the number of the units can be 4 or 5 without losing generality and an existing index structure can be constructed for the features.

5.3 Training the Hybrid Dimensionality-Reducer with Multi-Features Vectors

For multi-visual image features, the components in different feature vectors have different physical measurements. Each feature vector set has its correlation matrix. It is not suitable to use a PCA on this combination of visual features directly. Instead, the PCA is performed on each feature vector set separately. Thus for each image visual feature, a new data set of the original feature vectors can be obtained by this linear transformation. This new data set actually consists of the principal components of feature.

The neural network can be trained with a single feature or a combination of multi-features. This is due to the fact that the result of the network learning can be accumulated in the network's weights. If the network is trained by a single feature, each feature is presented to the network one after another. After the completion of each feature training, the weights contain the knowledge of that feature. The next feature training will be based on the results of the previous trained features. The network is tuned to combine the properties of current features with the previous training features. When all the features are presented to the network, the network integrates the knowledge of all the features into the weights of the network.

To start training the network, the weights of the network are initially set to a random value among [0,1]. Then a training sample containing the first feature of the image is presented to the network. The weights of the result of the first feature training are saved. These weights are then used as the initial weights for the training of the second feature. In general, the weights of the current training result are used as the initial weights of the next feature training.

The order in which the features are presented to the network for training is not important. However, the order of the feature may affect the rates of convergence of network training. If some features are more sensible to the human visual perception, the network training starts from these features may improve the rates of convergence of the neural network. This is due to the fact that these features are less ambitious and the number of iterations for the network is less. For example, the colour feature is more sensible to human perception than the texture feature (*i.e*, it is easier for a human to identify the colour of an image than its texture). The network training starts with the colour first and then with the texture. This can result in a fast convergence of the network.

Alternately, the network can be trained by a combination of multi-features. When the feature vectors of the image after the PCA analysis are merged as a new feature vector, the dimension of this new feature is equals to the sum of the dimensions of all of these features. Since the new feature vector consists of all the related features, it should also contain all the knowledge from these features. The training begins with initial random weights. A sequence of new feature vectors of images is presented to the network. Using this new vector to train the network, the weights of the network can be adjusted according to all the features at the same time. Once the network converges, the weights accumulate the knowledge about all these features.

The advantage of using a combination of multi-features to train the network is that it is simple and straightforward. It only needs to be given a random initial weight and then the network can be trained in the same way as a conventional neural network. If the network training samples are selected with care, the network can reach a global minimum. Single feature training is more complex. The network needs to save all the weights for each feature in order to perform the next step of the training. The extra computation of this operation can be costly compared to normal network training. In addition, the network needs to be converged several times. If there is one feature training running into a local minimum, the rest of the training can all fall into local minimum as well. It is for this reason that we chose the combination of multi-feature training methods to train the network rather than single feature training.

5.4 The Hybrid Training Algorithm for Dimension Reduction

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

Step 1: Find the correlation (covariance) matrixes of the all images for each different visual feature.

Step 2: For each correlation (covariance) matrix of the feature, compute the correspondent eigenvalues and arrange them based on the descending order of their values.

Step 3: Select the first *m* eigenvalues from each list of eigenvalues in Step 2 by using a cut-off value λ. This cut-off value accounts for the total variance of these *m* eigenvalues.

Step 4: Find the *m* eigenvectors of each correlation (covariance) matrix and rearrange them based on the descending order of their corresponding *m* eigenvalues into the rows of the PCA transformation matrix T.

Step 5: Obtain the new representation Y_i for each image features X_i by using the PCA transformation:

$$Y_i = T\,X_i \qquad\qquad i = 1, ..., m.$$

Step 6: Select the training samples from the image database. Classify these training samples into different classes based on the human visual perceptions.

Step 7: Prepare the training pattern for the classifier.

Step 8: Set all the weights and node offsets of the network to small random values.

Step 9: Present the training samples as input to the network. The training sample could be new on each trial or samples from a training set could be presented cyclically until the weight stabilizes.

Step 10: Use the quickprop learning algorithm, which was already described in Chapter 4, to update the weights of the network.

Step 11: Test the convergence of the network. If the condition of convergence of the network is satisfied, stop the network training. Otherwise, go back to Step 8 and repeat the processing.

Step1-5 are an unsupervised learning. The PCA is performed on all images rather than the training samples set only. The advantage of using the PCA on all of the images rather than on the training set is that the global correlation (covariance) matrix contains all the information about the images. Thus the information in the training set based on these matrixes can better represent all the images. In addition, in the dimension reduction phase, the input to the reducer for each image is the reduced dimension of the PCA rather than the original dimension input. This can improve the performance of the network training.

The number of principal components is determined by the cut-off value λ. There is no formal method to decide the threshold of this cut-off value. In Step 3, the cut-off value λ is set to 0.1 in order to discard any eigenvalue which has the zero-variance contribution to the total variance and to keep the first m eigenvalues with non-zero variance. The assumption for this setting is that all the eigenvectors corresponding to value λ will not make a significant contribution to the network training. It is noted that the value of m for each feature is different. The value of m may be one-third or one-half of the dimensions of the original input space. It is acceptable in our case since result of the PCA transformation is not directly used for the representation of the image but for the next step of the non-linear dimension reduction. It is desirable to keep the minimum principal components required without losing too much information. With this method, if a new image is inserted into the image collection, a PCA is required to re-compute the correlation matrix on the modified collection. However, recently, a modified PCA [Chandrasekaran *et al.*, 1996] has been proposed to overcome this problem. With this modified PCA, any change to the image collection will not require the re-computation of the correlation matrix.

After the PCA analysis, the new representation of the original image collection is formed. The selections of the training samples are conducted by the domain experts who can better identify image similarities and classify the training samples into corresponding classes based on the similarity of each feature. Each class of image is similarly based on all the features which are required to retrieve images from the image database. Clearly the quality of the human classification of images into their corresponding classes will greatly affect the performance of network training.

For each training sample, a corresponding pattern pair is constructed in Step 7. Each pattern pair contains a continuous valued vector and its desired outputs. All the desired outputs are typically set to zero except for that corresponding to the class in which the input is formed. The desired output is set to 1. The vector is the combination of all features of that image in the form of the representation Y_i in Step 6. The order that the features appear in this vector is not important.

The network training starts at Step 8-11. In general, the weight of network is randomly set to a small continuous value initially. The network learns by the "learning by epoch" approach. This means that the weights' updates only happen after all the training samples have been presented to the network. In the quickprop learning algorithm, the parameter v which 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. The learning rate is not changed during network training. In every iteration of training, the error surface 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 eigenvalues for each feature. These largest eigenvalues have the largest role in the classification of the pattern. Thus the network has less chance of being trapped in a local minimum.

The condition of network convergence is indicated by the total gradient error or the total bit errors. When these two parameters do not change in the 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 bit errors since the network also function 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 increases to 1.

It is obvious that this hybrid dimension reduction for images is computationally more efficient than the standard neural network with the original dimension data input. It is because hybrid dimension reductions use a relatively small number of network inputs. In addition, the network training iterations are conducted in the direction of the largest eigenvalues for each feature only.

5.5 The Application of Image Feature Dimension Reductions

In this section, we use nine images to illustrate the performance of this hybrid method. The feature vectors of the nine images are the same as those used in Chapter 3 and Chapter 4 which are derived from the nine images from Appendix 1. We used two features of each image namely colour (1 - D) and texture (30 - D). The PCA is first performed on the texture of nine images to achieve the first linear dimension reduction. The eigenvalues and the percentage of total variation are the same as in Table 3.6.1 in Chapter 3. The first 10 eigenvalues in the descending order account for 100% of the total variation for these nine images. If the cutoff value is 0%, the rest of the eigenvalues after the 10^{th} can be considered not to make a significant contribution to the classification task and can be excluded from the network training set. Based on these eigenvalues, we can obtain 10 eigenvectors corresponding to these eigenvalues. These eigenvectors then become the intermediate representation of the image feature vectors. Therefore, after the linear dimension reduction, the 30 dimensions of the texture vector are reduced to 10 dimensions without losing the maximum variance among the images.

It should be noted that since the vector of the colour feature in our example is only a one-dimension vector, it is not necessary to perform the PCA on it. This one-dimension vector of colour can be used directly for the nonlinear dimension reduction phase. Thus a new combination vector of the texture and colour features for each image is obtained by merging these two feature vectors. The first 10 components of this new combination vector represent the texture and the last component of this new vector represents the colour feature. The order of these two feature vectors is not important since the weights of the network are updated according to two features at the same time. However the order of components of each feature is important. The order of the components in the same feature vector should be arranged in the descending order of their corresponding eigenvalues. The complete training set of these nine images is given in Table 5.5.1. Each image corresponds to a row in the table which consists of 10 components of the texture feature and one component of the colour feature. The symbol *ti* denotes i*th* component of the texture feature which is in the descending order of their total variance, which c1 denotes the colour feature component.

Table 5.5.1 Combination representations of colour and texture for nine images

	Texture feature components										Colour
Image	t1	t2	t3	t4	T5	t6	t7	t8	t9	T10	c1
1	-0.782	0.019	-0.092	-0.459	0.275	0.1653	0.5926	0.276	0.0000	0.0000	0.65555
2	-1.118	0.499	-0.273	0.348	0.454	-0.4487	-0.0253	0.014	0.0001	0.0000	0.64434
3	-1.166	-0.290	0.3437	-0.432	0.385	0.2133	-0.4899	0.234	0.0005	0.0004	0.64444
4	0.356	-1.320	1.2533	0.088	-0.140	-0.1364	0.1132	-0.238	0.0002	0.0000	0.59167
5	-0.105	-0.910	-0.962	-0.749	-0.560	-0.1032	-0.0796	-0.189	0.0000	0.0000	0.59722
6	1.905	-0.850	-0.602	0.733	0.349	0.0959	-0.0400	0.248	0.0000	0.0000	0.58334
7	-0.208	0.825	0.2413	0.474	-0.780	0.0055	-0.0392	0.574	0.0002	0.0000	0.33333
8	1.951	1.473	0.2692	-0.677	0.126	-0.0338	-0.0513	-0.197	0.0000	0.0000	0.33322
9	-0.833	0.558	-0.179	0.673	-0.110	0.2422	0.0195	-0.722	0.0005	0.0004	0.20833

After the combined representation vectors of colour and texture for these images is obtained, the nonlinear dimension reduction can be carried out on this reduced dimension of vector data set. The training set for the network can be obtained by replacing the v(*i*) in the Table 4.5.1 in Chapter 4 with the corresponding row *i* in the Table 5.5.1. The output and the image classes for the above Table 5.5.1 are exactly the same as in Chapter 4 since the images are the same in both cases.

The network is trained by the quickprop algorithm where the learning rate is set to 0.9 and the step-size parameter v is set to 1.75. The weights of the network were randomly initialized within the range [0, 1] and the activation of the network is a nonlinear asymsigmoid function. Table 5.5.2 shows the result of the 2-dimensional representation of these nine images after this hybrid dimension reduction.

Table 5.5.2 2-D representations of combination of colour and texture.

Image	PC1	PC2
1	0.820702	0.387839
2	0.830197	0.453887
3	0.832931	0.466538
4	0.24225	0.835808
5	0.315795	0.863584
6	0.155715	0.873646
7	0.092241	0.074006
8	0.031459	0.039013
9	0.073881	0.096971

To visualize the result of this 2-D representation, we plot this data set in a two-dimension space as shown in Figure 5.5.2. It can be seen that the three classes of nine images are classified very well. The difference between Figure 5.5.2 and Figure 4.6.1 in Chapter 4 lies in different mapping. Each class in Figure 5.5.2 is closer together than in Figure 4.6.1. Some data point even overlap such as *i2* and *i3*, due to the fact that some information has been lost during the dimension reduction in the first stage of PCA. In addition, the data points in Figure 5.5.2 represent not only the texture feature but the colour feature as well. Thus the weighting of various features of the image based on human visual perceptions can now be represented as a mathematical similarity. Image similarity retrieval based on the combination of texture and colour becomes the measurement of the distance between the points in this 2-dimension feature space.

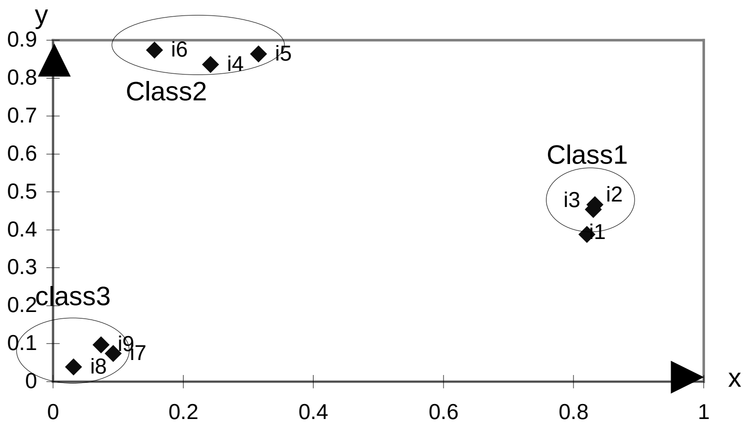


Figure 5.5.2 Plot of nine images with hybrid dimension reduction

5.6 Summary

The dimension reduction of image features is a highly nonlinear task. Although a linear PCA is simple and straightforward, it can not reflect the knowledge of image feature weighting which is unequal in human visual perceptions. On the other hand, neural network training for NLDR can be slow, when the input dimension is very high. In this chapter, we have proposed a new hybrid method of dimension reduction for image features. The architecture of this hybrid method has been presented and the hybrid training algorithm is discussed as well.

Hybrid Dimension Reduction

The result of applying hybrid methods for dimension-reduction to the same nine images is given. We compare the results of dimension reduction using hybrid methods versus standard neural networks. It has been shown that the input dimensions of the network for the NLRD are reduced to one-third of the original dimensions of the input, but this hybrid method can still perform classification tasks as well as the standard neural network does. In next chapter, we will show that this hybrid method of NLRD is more efficient than the standard NLRD.

Chapter 6

Experiments and Discussions

The purpose of this thesis is to investigate an efficient method for dimension reduction of visual features of images in order to index multi-features in image collections. We have described three methods for dimension reduction of multi-visual features in previous chapters, i.e., PCA, Neural Network and Hybrid PCA Neural Network. This chapter presents three experimental results. The aim of these experiments is to demonstrate that hybrid dimension reduction is superior to PCA and neural networks. The first experiment shows that the result of using a PCA for the dimension reductions of multi-feature images. The second experiment evaluates the neural network for dimension reduction and the third experiment justifies the efficiency and accuracy of the hybrid dimension reduction we proposed.

6.1 Experimental Conditions

6.1.1 Test Image Collection

We use a small research image collection which consists of over 350 images. These images are taken from different public domains that can be classified under a number of themes. These collections include nature scenes, flowers, rocks, buildings and general images. The definition of the image similarities in two features is not defined by any formal mathematical similarity function. Instead, the similarity of each image is decided by human visual perceptions. Several people with different backgrounds helped to classify these images into their corresponding classes based on their similarity of colour, texture, and combination of colour and texture. If the majority of people consider the images from the collection are similar, then these images can be considered to be in the same class. Since there are many factors that can affect human visual perceptions as discussed in Chapter 2, the domain expert can perform the image classification much better.

Table 6.1.1 shows the ten groups of images from this collection, which are used in the following experiments. Images in class1 to class7 are similar in both colour and texture and are used for training and testing the dimensionality-reducer. Images in class 8 are similar in both colour and texture and are for testing the dimensionality-reducer but not for training it. In class 9 and class10, the images are either similar in colour or texture. These images are used for testing the performance of the dimensionality-reducer.

Table 6.1.1 Collections of images used as a test bed

Class of images	Number of images	Description	Function
Class1	5	Natural objects such as sky star, similar in colour and texture.	PCA, network training and testing
Class2	7	Various images of sunset scenery similar in colour and texture.	PCA, network training and testing
Class3	9	Various images of human face similar in colour and texture.	PCA, network training and testing
Class4	10	Various images of starts similar in colour and texture.	PCA, network training and testing
Class5	7	Various images of balls similar in colour and texture.	PCA, network training and testing
Class6	6	Various images of decoration materials similar in colour and texture.	PCA, network training and testing
Class7	9	Various images of fireworks similar in colour and texture.	PCA, network training and testing
Class8	10	Various images of road signs similar in colour and texture.	Neural network testing
Class9	20	Various images similar to images of Class1-7 in colour only.	Neural network testing
Class10	20	Various images similar to images of Class1-7 in texture only.	Neural network testing

6.1.2 Image Features

In most CBIR systems, images are presented by their visual features vector. Image features used in these experiments are colour and texture. These features of image in Table 6.1.1 are extracted by using the following feature extraction methods.

Experiment and Discussion

Several methods can be used to extract colour from images. In these experiments, we take the images in the *ppm* format (each pixel colour is represented as an (R, G, B) tristimulus), then convert the (R, G, B) triple of each pixel of images into the corresponding hue value in the range [*0,1*] by using the following formula:

$$H = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right\}$$

If B > G, then H = 1- H/2π, otherwise, H = H/2π.

When the hue for each pixel in the image is calculated, the corresponding bin in the histogram is incremented. The modal bin is found from the resulting histogram and it is taken as the feature of the image. If G = B = R, the above formula is undefined. To solve the problem of hue unsaturated, we use *I* (intensity) to replace the hue when unsaturation condition occurs. *I* can be defined as:

$$I = \frac{R+G+B}{3}$$

Thus a colour feature vector is a one-dimension vector, since both values are one-dimension value. The detail of the implementation of colour feature extraction is described in Tam [1995]. Some examples of colour extraction using this method are given in Appendix 1.

The texture feature in this experiment is extracted using a Gabor filter based method as a 30-dimensional feature vector. This method makes use of amplitude spectra of images. It detects the global periodicity in the images by identifying high energy narrow peaks in the spectrum. It requires a set of selected orientations and spatial frequencies. The equation of Gabor filter can be represented as:

$$g(X,Y) = \exp\{-4\ln 2[(\frac{P_x}{a})^2 + (\frac{P_y}{b})^2]/2w^2\} * \exp[2\pi fj(x\cos\theta + y\sin\theta)]$$

where θ is the orientation, *f* is the spatial frequency and *w* is the bandwidth, *a* and *b* are the width and height of the filter; and

$$P_x = x\cos\varphi + y\sin\varphi; P_y = -x\cos\varphi + y\sin\varphi$$

The advantage of the Gabor method is the consistent interpretation over both natural and artificial images. The implementation details of texture extraction can be found in Tiu [1995]. An example of an image and its corresponding texture vector extracted by this method is shown in Appendix 1.

6.2 The Benchmark of the Experiment

The aim of these experiments is to determine the accuracy and efficiency of the three methods of dimension reduction. The images are represented by their corresponding feature vectors (31-dimensions: colour 1-dimension; texture 30-dimension) which can be viewed as points in the feature space. Thus the distance between any two points in this feature measures the similarity of two images. After dimension reduction of the image feature, a new feature space that combines colour and texture is formed. The new feature space contains the similarity information of the image in both colour and texture. The distance between two data points in this space represents the visual similarity of their original images in colour and texture. In order to visualize the similarity of image in the unique feature space, we map this results of dimension reduction methods into a 2-dimension new feature space and observe the data cluster directly.

The learning time parameter of *t* is used to indicate the efficiency of the dimension reduction, that is, the total epochs required for training the dimensionality-reducer. It is noted that the PCA is performed by the SVD and is not used for the final dimension reduction. Thus we will ignore this parameter for the PCA.

6.3 Experimental Results of Dimension Reduction

6.3.1 Result for Principal Component Analysis

In this experiment, the PCA is performed on image class 1 to class 7 that are both similar in colour and texture. It should be noted that the feature vectors of these images are combinations of colour and texture vectors. Table 6.3.1.1 shows the eigenvalues and the percentage of total variation explained. The eigenvalues are in descending order, *i.e.*, $\lambda_1 > \lambda_2 > \ldots > \lambda_{31}$. The first ten eigenvalues explain *94.1%* of the total variance and should be sufficient to analyze the influence of the image features. If we choose the first ten eigenvalues and calculate the corresponding ten PCs, then the *31* dimensions of the image feature data set are reduced to an 11-dimension feature data set. The corresponding ten PCs of the image collection are shown in Table 6.3.1.2.

Table 6.3.1.1 The eigenvalues and the percentage of total variation

No.	1	2	3	4	5	6	7	8	9	10
λ	10.03	6.29	5.32	2.20	1.52	1.16	0.90	0.64	0.60	0.52
%	32.3	20.3	17.2	7.1	4.9	3.7	2.9	2.1	1.9	1.7
No.	11	12	13	14	15	16	17	18	19	20
λ	0.349	0.280	0.246	0.207	0.181	0.124	0.108	0.070	0.063	0.056
%	0.1	0.9	0.8	0.7	0.6	0.4	0.3	0.2	0..2	0.2
No.	21	22	23	24	25	26	27	28	29	30-31
λ	0.04	0.03	0.02	0.01	0.01	0.01	0.01	0.00	0.00	0.00
%	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0

To visualize the result of this linear transformation, we selected the first two PCs in Table 6.3.1.1 and plotted the data set in Figure 6.3.1.1 with respect to these two dimensions. The first two PCs for the above data set account for *32.3%* and *20.3%* of the total variation respectfully. Thus, a two-dimensional plot with respect to these PCs, representing 52.6*%* of the variation, will give a reasonable approximation of the relative positions of the observations in the *31*-dimensional space.

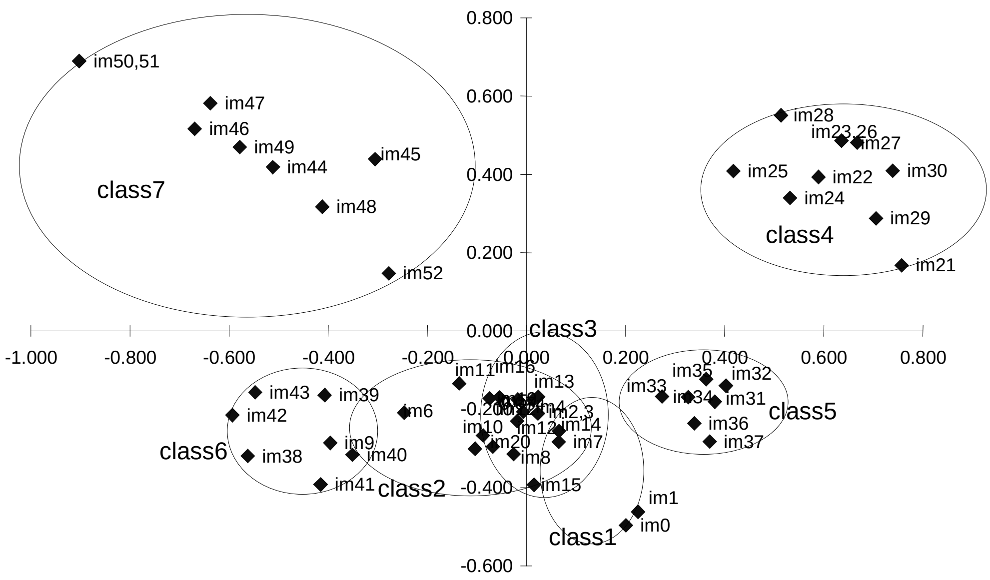


Figure 6.3.1.1 Plot of collections of image with respect to their first two PCs

There are seven circles that denote the seven classes of images in Figure 6.3.1.1. Classes 4,5 and 7 do not overlap with any other class of image and can be considered to maintain the similarity of image after the PCA transformation. Class 6 slightly overlaps with Class 2 (im9, im41). Classes 1,2 and 3 overlap intensely around the Class3 area. If any distance function is applied directly to Classes1-3, the distance of the images among each class may be greater than the distance of images between two different image classes. For example, the distance of observation *im4* and *im1* that are in the same class is greater than the distance of observation *im4* and *im7* that are not in the same class. Thus *im4* should be considered more similar to *im7* than *im1*, but this is not true. It indicates that the similarity of images in those classes can not be maintained after the PCA transformation. Image retrievals based on this method can result in the problem of false dismissal (*i.e.*, all qualifying images may not be retrieved).

6.3.2 Result for Neural Networks

In order to overcome the problem of the PCA, in this experiment, we use a three-layer neural network discussed in Chapter 4 to reduce the feature dimension of the images collection in Table 6.1.1. All vectors of image feature are 31 dimensions vectors which contain colour and texture features. The network training set contains image feature vectors from Classes 1-7. The network testing data set is drawn from the image collection in Table 6.1.1. This includes some training samples and non-training samples. The recognition rate is defined as the percentage of test images that the network can recognize. The rejection rate is defined as the percentage of images correctly rejected by the network. The learning rate is set to 0.9 and the maximal growth factor is set to 1.75 in the quickprop learning algorithm. The weights were chosen randomly within [0, 0.7] range. Table 6.3.2.1 shows the accuracy of the results of the network training.

Table 6.3.2.1 Results of the network training

Image class	Recognition rate	Rejection rate
1	100	0
2	100	0
3	100	0
4	100	0
5	100	0
6	100	0
7	100	0
8	10	90
9	5	95
10	2	98

The learning time is reported as the average number of epochs required until the network converges. The network convergence can be measured by the total error or total error bits of the network. Figure 6.3.2.1 shows the time of the network learning with two hidden units.

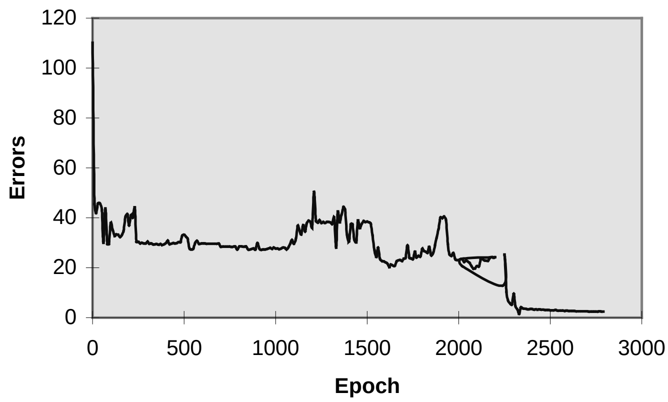


Figure 6.3.2.1 Learning time of neural network with two hidden units

When the network learning is about 2500 epochs, the errors of the network tend to be steady at about 0.2. It is not necessary to get the zero error since the error of 0.2 is already small compared to the initial error. Thus the network learning time *t*, is 2500 epochs. After the network training is completed, dimension reduction can be achieved by feeding the image feature vectors into the network and taking the value of the hidden units as the lower dimension representation. Figure 6.3.2.2 shows a 2-dimensional result of the image Classes1-7 by using this network.

In Figure 6.3.2.2, it can be seen that seven classes of the image collection can be separated very well in the two-dimension feature space. The distance of almost any two observations in one class is less than the distance of any two classes. Thus image retrieval based on this method can overcome the problem of false dismissal which is the problem in PCA. However, in Figure 6.3.2.1, it can be seen that the learning time is very long. In next section, we show that the hybrid method can improve the network learning time without losing the accuracy.

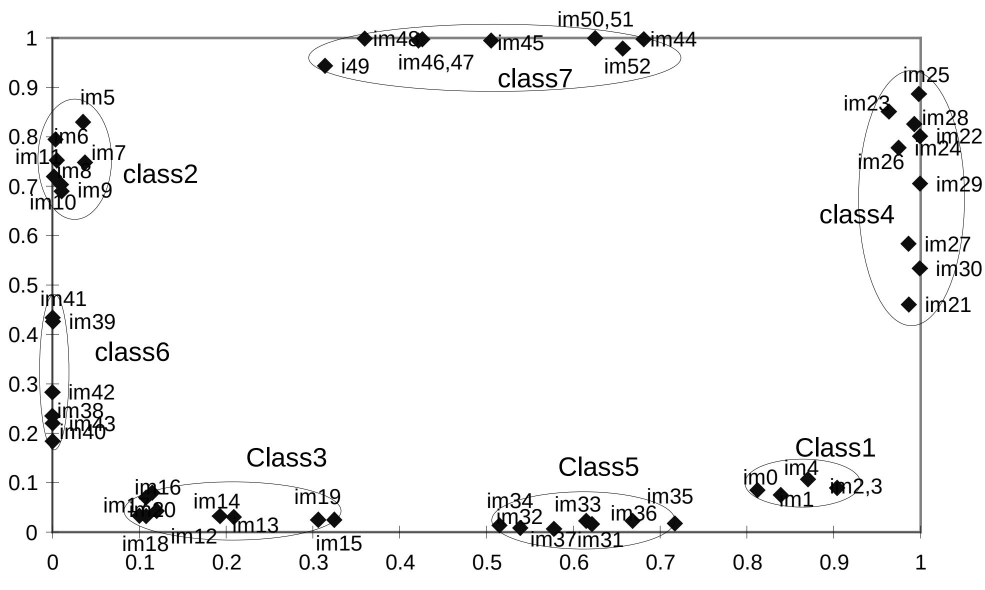


Figure 6.3.2.2 Plot of the collection of images using the network approach

6.3.3 Result for Hybrid Dimension Reduction

In this experiment, we apply the hybrid dimension reduction to the image collection in Table 6.1.1. The use of the PCA first reduces the dimension of the network training samples. We select the first 10 PCs of image texture, which account for about 94.7% of the total variance as the representation of the texture. Thus the inputs of the network are reduced to 11 dimensions (10D for texture and 1D for colour). The network training condition of this hybrid dimension reduction is the same as described in section 6.3.2. Table 6.3.3.1 shows the accuracy of the result of the hybrid network training and Figure 6.3.3.1 shows the time of the hybrid network learning with two hidden units.

Table 6.3.3.1 Results of the hybrid network training

Image class	Recognition rate	Rejection rate
1	100	0
2	100	0
3	100	0
4	92	8
5	94	6
6	100	0
7	100	0
8	12	88
9	10	90
10	5	95

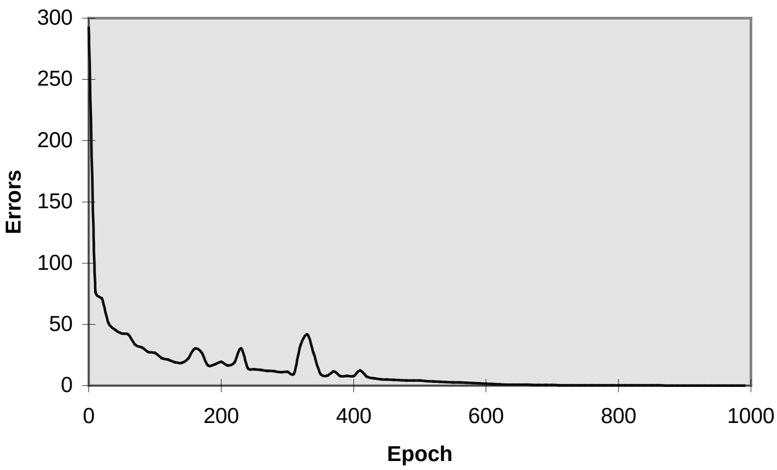


Figure 6.3.3.1 Learning time of the hybrid dimension with two hidden units

When the network learning time reaches 550 epochs, the errors of the network are steady at about 0.02. It indicates that the network learning will be successful after 550 epochs. Thus the hybrid network learning time *t* is 550 epochs. It is about 5 times faster than the standard network training. Figure 6.3.3.2 shows a 2-dimension result of image Classes1-7 by using this hybrid learning algorithm.

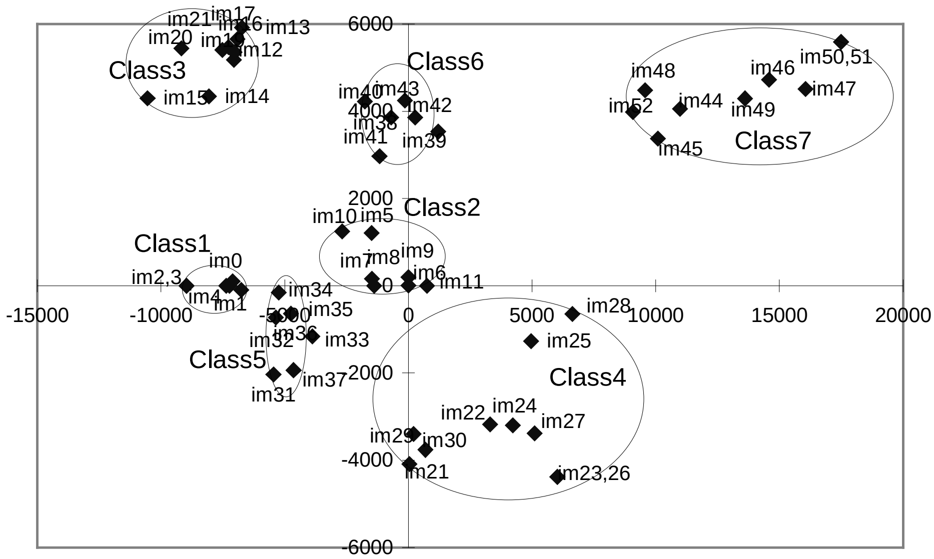


Figure 6.3.3.2 Plot of the collection of images using the hybrid dimension reduction

In Figure 6.3.3.2, the seven classes of the image collection can be separated in the two dimensions feature space. The distance of almost any two observations in one class is less than the distance of any two observations in different classes except for images in Class 1 and Class 5. Distance of *im31* and *im34* in Class 5 is longer that the distance of *im34* and *im1*, *im2* and *im3* of Class 1. It should be noted that images in Classes 1, 2 and 3 are closer and even overlap. This is because some information is being lost during the PCA reduction. That could be a disadvantage for doing the images exact match retrieval since images are not identical in this unique feature space. However, this is not the problem for image similarly retrievals. Thus image retrieval based on this method can overcome the problem of false dismissal.

6.3.4 Evaluation of Reduced Dimension Image Features Using R^+ -tree

Here we use an R^+ tree for evaluating the result of indexing using a combination of different features whose dimensions have been reduced by various dimension reduction methods. The dimensions of image features are reduced to five dimensions from the original 31 dimensions using the above dimension reduction method. This dimension can be handled very well by the R^+ tree. Every image in each class is a query image. We pose the query images to the R^+ tree and retrieve all images that contain a set of images, which match the query within a tolerance *t*. The tolerance *t* is defined as the number of images in each class. False dismissal means that qualifying images may not be retrieved. This value is the average of false dismissed images returned for each class. It can be calculated by the following:

$$\text{False dismissal} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{E_i}{N}\right)$$

where N is the total number of query in each image class, and E_i is the number of false dismissal images returned by i*th* query image.

False alarms refer to no-qualifying returns in which it may return a subset of the required images. It is an average of the queries of no-qualifying images returned by each class and is defined as follows:

$$\text{False alarms} = \frac{E_q}{N}$$

where E_q is the total number of returned queries that contain no-qualifying images and N is the total number of queries for each image class. Table 6.3.4.1 shows the results of queries of these seven classes.

Table 6.3.4.1 Results of retrievals using a R^+ tree

	PCA		Neural Network		Hybrid Method	
Image class	False alarm (%)	False dismissal (%)	False alarm (%)	False dismissal (%)	False alarm (%)	False dismissal (%)
1	100	40	0	0	0	0
2	100	40.1	0	0	14.2	3
3	44.4	12.4	0	0	0	0
4	0	0	0	0	0	0
5	42.9	12.3	0	0	0	0
6	50	19.5	0	0	0	0
7	33.3	18.6	0	0	0	0
Average	27.2	20.1	0	0	2	0.428

The result of Table 6.3.4.1 shows that dimension reduction of the PCA can cause either false alarm or false dismissal for all classes except Class4. This is the same as the result observed in Figure 6.3.1.1. However, the dimension reduction of the neural network has neither A false alarm nor A false dismissal. The dimension reductions using hybrid methods have 2 percent false alarm and 0.428 percent false dismissal images in the total queries. This is not much different with dimension reductions in neural networks. Thus the major difference of the two approaches is the time needed to train the network. Hence, it is advantageous to use a hybrid dimension reduction to reduce the image features for indexing with a R^+ -tree.

6.4 Analysis and Discussion

The above experimental results have shown that the hybrid dimension reduction is superior to the other two dimension reduction methods. In this section we present a discussion of the issues related to the performance of this hybrid method.

6.4.1 The Importance of Selected Network Training Set

The training set used for the network training plays an important role in the system. The network performance is greatly affected by the people who train the network. There are many factors affecting human visual perceptions that we have already discussed in Chapter 2. The training set used for the neural network is selected based on human visual perceptions. There are not any formal methods defined to select the training set and currently it is chosen manually. It is laborious for a large image collection. A domain expert together with various data analysis techniques can help to improve the selection of the training set. The image feature vector can be extracted by robust image processing techniques and analyzed by data analysis techniques such *k*-means clusters that are used to arrange the images based on the similarity of each individual feature. The intersection of these results can become the initial training set. Then the domain expert can be employed to fine-tune this initial training set.

6.4.2 Parameters of the 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 experiment. However, in practice, it is often undesirable or even impossible to perform 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 the maximal growth factor of 1.75 and the learning rate 0.9.

The number of the hidden units used can also affect the network convergence and learning time. The more the hidden units, the easier the network learns. This is because more hidden units can keep more information. However, the network is a dimension reducer, thus the number of hidden units are restricted to a practical limit. For example, if we use 5 hidden units in section 6.3.3 then the learning time can reduce dramatically and the network can even reach zero error. Figure 6.4.2.1 shows the learning time of hybrid method with 5 hidden units. It takes only 40 epochs to reach an error of 0.02 compared to Figure 6.3.3.1 in which it takes about 550 epochs.

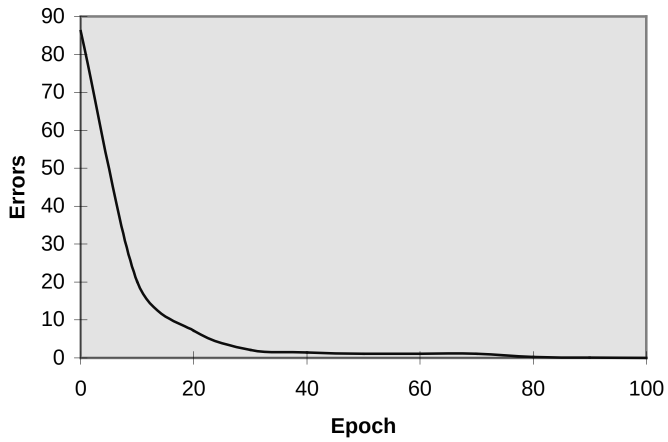


Figure 6.4.2.1 Learning time of hybrid dimension with five hidden units

6.4.3 Principal Components Selected for Network Training

In the hybrid dimension reduction, the input of the network is not the dimension of original features set but the output of the PCA. The number of the 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 very well for few PCs since some information may not be included in the network training. We have discussed the theoretical issues involving the selection of the PCs in Section 3.7.1. In this section, we give the results of using different PCs for the hybrid dimension reduction for the image collection in Table 6.1.1. The network training condition is the same as in section 6.3.3 with 5 hidden units. Table 6.4.3.1 shows the learning time for different PCs.

Table 6.4.3.1 Learning time for different PCs

Number of PCs	Total variation (%)	Learning errors	Number of epochs
2	52.6	4.41	>2000
3	69.8	0.02	240
5	81	0.02	30
10	94.1	0.02	40
15	98.2	0.02	60
20	99	0.02	65

It can be seen that the numbers of PCs for the best network training in our application depend on their total variation. There is not a significant difference in the time required for network training from 5 to 20 PCs since their variations account for more than 80 percent of the total variation. Moreover, the PCs are in the descending order of their variations. Increasing the number of PCs after the first 5 PCs does not require much extract time to train the network. For example, there are only 5 epochs difference between 15 PCs and 20 PCs. However, if we choose the number of PCs with variations less than 70 percent of the total variation, there are significant differences. It takes more than 2000 epochs for 2 PCs which account for 52.6% of the total variation to reach the ultimate network error of 4.41 which is far greater than the normal 0.02 error.

Table 6.3.1.2 The first ten PCs of image collection

image	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
im0	0.201	-0.496	-0.163	0.193	-0.013	-0.457	-0.008	0.025	0.179	0.045
im1	0.225	-0.462	-0.171	0.156	-0.020	-0.434	-0.020	0.020	0.175	0.074
im2	0.023	-0.211	-0.880	0.344	0.048	0.366	0.001	0.021	0.084	-0.067
im3	0.023	-0.211	-0.880	0.344	0.048	0.366	0.001	0.021	0.084	-0.067
im4	-0.007	-0.207	-0.707	0.135	0.053	0.007	0.313	0.209	-0.340	0.060
im5	-0.018	-0.175	0.101	-0.132	0.285	-0.041	0.258	-0.104	-0.138	0.085
im6	-0.247	-0.209	0.485	0.291	0.203	0.023	0.009	0.255	-0.092	-0.164
im7	0.065	-0.284	0.219	-0.009	0.295	-0.138	0.244	-0.236	0.005	0.018
im8	-0.026	-0.314	0.341	0.179	0.135	-0.331	-0.115	0.231	-0.137	-0.005
im9	-0.396	-0.286	0.517	0.415	-0.094	0.189	0.205	-0.065	0.192	0.005
Im10	-0.104	-0.301	0.058	-0.051	0.250	-0.020	0.327	-0.225	-0.081	-0.010
Im11	-0.135	-0.135	0.461	0.227	0.249	0.087	-0.024	0.122	-0.052	-0.209
Im12	0.014	-0.177	0.021	-0.422	-0.091	0.065	0.007	0.078	0.002	-0.103
Im13	0.024	-0.168	-0.115	-0.400	-0.083	0.037	-0.020	0.027	0.010	0.003
Im14	0.066	-0.256	-0.048	-0.317	-0.006	0.071	0.089	-0.075	0.149	-0.124
Im15	0.016	-0.393	-0.273	-0.251	-0.126	-0.138	0.147	0.075	-0.010	-0.087
Im16	-0.054	-0.170	0.032	-0.455	-0.054	0.084	-0.026	0.054	-0.029	-0.061
im17	-0.074	-0.173	-0.097	-0.425	0.057	0.101	0.042	0.063	-0.005	0.025
im18	-0.088	-0.267	-0.091	-0.315	0.070	0.088	-0.025	-0.064	0.138	-0.040
im19	-0.019	-0.231	-0.209	-0.303	-0.024	0.027	-0.103	0.244	0.098	0.054
im20	-0.068	-0.296	-0.111	-0.379	-0.024	-0.041	-0.004	-0.047	0.000	0.014
im21	0.757	0.168	0.224	0.055	-0.054	-0.069	-0.062	0.000	-0.091	-0.021
im22	0.590	0.393	-0.019	0.053	0.252	0.021	0.037	0.120	0.147	0.285
im23	0.636	0.486	0.312	0.058	0.001	0.046	0.101	0.023	-0.067	0.074
im24	0.532	0.340	0.258	0.065	0.056	0.113	0.022	-0.109	0.011	-0.096
im25	0.417	0.409	0.219	-0.063	0.002	0.068	-0.033	-0.018	0.011	-0.020
im26	0.636	0.486	0.312	0.058	0.001	0.046	0.101	0.023	-0.067	0.074
im27	0.668	0.481	0.315	-0.052	-0.020	0.102	-0.004	-0.035	-0.070	-0.009
im28	0.514	0.551	0.077	-0.095	-0.086	-0.121	-0.151	0.025	0.132	-0.133
im29	0.705	0.288	0.103	0.061	0.000	-0.015	0.119	0.078	0.172	-0.055
im30	0.739	0.410	-0.006	0.069	-0.232	0.056	0.131	0.077	0.030	-0.150
im31	0.380	-0.181	-0.143	-0.041	-0.156	-0.060	-0.092	0.060	-0.147	0.029
im32	0.403	-0.140	-0.150	0.030	-0.201	-0.022	-0.188	-0.022	-0.135	0.042
im33	0.274	-0.167	-0.133	0.150	-0.051	0.091	-0.099	-0.169	0.005	0.043
im34	0.327	-0.169	-0.103	-0.028	-0.107	0.010	-0.087	-0.089	-0.102	0.052
im35	0.363	-0.123	-0.119	-0.006	-0.094	0.014	-0.065	-0.060	-0.116	0.043
im36	0.339	-0.236	-0.298	0.240	-0.128	-0.068	-0.129	-0.175	-0.024	0.049
im37	0.370	-0.283	-0.236	0.258	-0.197	-0.091	-0.150	-0.179	-0.004	-0.014
im38	-0.562	-0.320	0.204	0.121	-0.133	0.111	-0.224	-0.047	-0.197	-0.006
im39	-0.407	-0.164	0.386	0.040	-0.010	0.059	-0.146	0.010	-0.013	-0.025
im40	-0.351	-0.316	0.390	-0.060	0.068	0.036	-0.107	-0.083	0.031	-0.145
im41	-0.415	-0.393	0.426	0.184	0.074	-0.025	-0.036	-0.036	0.088	-0.080
im42	-0.593	-0.215	0.465	0.092	-0.271	0.124	0.033	0.070	-0.003	0.174
im43	-0.547	-0.157	0.591	-0.003	-0.411	0.233	0.049	0.056	0.072	0.308
im44	-0.511	0.419	-0.253	0.059	0.271	-0.032	-0.204	0.083	-0.007	-0.049
im45	-0.305	0.439	-0.088	-0.082	0.262	0.065	-0.147	-0.033	0.012	0.070
im46	-0.669	0.517	-0.327	-0.005	0.101	-0.087	-0.045	-0.122	-0.042	0.086
im47	-0.638	0.582	-0.089	0.020	0.088	-0.068	-0.115	-0.081	0.022	-0.030
im48	-0.412	0.317	0.020	-0.134	0.079	-0.049	-0.094	-0.138	0.021	0.017
im49	-0.578	0.470	-0.126	-0.043	0.228	-0.034	-0.138	-0.055	-0.108	0.092
im50	-0.902	0.690	-0.231	0.074	-0.373	-0.190	0.207	0.007	0.002	-0.101
im51	-0.902	0.690	-0.231	0.074	-0.373	-0.190	0.207	0.007	0.002	-0.101
im52	-0.277	0.147	-0.240	0.026	0.254	0.013	0.009	0.186	0.208	0.150

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Currently there are several popular optimized SAM available for multi-dimensional indexing in multimedia databases. However, there are two major limitations for using any of these approaches. The first limitation is that a distance function is required to measure the distance of two image points in the feature space. The second limitation is that the dimensionality of the image features needs to be as low as possible. These limitations restrict their application in multimedia database systems. A query may use a combination of different sets of features that in turn can have a very high dimension. Moreover, it is difficult to have a distance function to measure the similarity of the combination of image features.

One contribution of this research is the introduction of an older statistical method, namely PCA, for reducing the image features. The PCA is a linear transformation technique therefore, it is much faster than nonlinear regeneration. Although the results of this approach are not suitable for indexing image features using SAM, it could be added to the arsenal of multimedia database research to help with the indexing and visualization of multimedia data sets. The PCA had been used in our proposed hybrid method of dimension reduction to provide the optimal solution for the selection of the best features to train the hybrid dimensional reducer.

Another contribution of this research is the introduction of artificial intelligence techniques into the multimedia database areas to index combined image features. The aim of this technique is to reduce the dimension of image features into a unique feature space similar to human visual perceptions. Thus any existing distance functions can be used as a similarity metric, such as Euclidean distance function. We have discussed different neural network architectures together with different network learning algorithms for the dimension reduction of image features. A three-layer feedforward network with quickprop learning was chosen for the NLDR of image features. This NLRD has achieved the goal of dimension reduction and the integration of combinations of image feature. It provides a guideline for dimension reduction for indexing in multimedia databases. Although this model is a nonlinear regression problem and the learning time may be high, when combined with PCA, it forms the backbone of our proposed hybrid dimension reduction.

The main contribution of our research is to propose a fast and robust hybrid nonlinear dimension reduction method for indexing of multiple visual features in image databases. This proposed approach combines a statistical approach PCA and a neural network technique to achieve the goal of dimension reduction of image features. We have described the architecture in which PCA is used as a preprocessor to reduce the dimension of input to the neural networks, a three-level feed forward neural network is then used to perform the dimension reduction of image features. We have developed a hybrid learning algorithm to train the hybrid dimension reducer. The hybrid learning algorithm is the combination of supervised and unsupervised learning algorithms. Therefore, this learning algorithm can compensate for the characteristics of slowness of the supervised learning and lower level accuracy of unsupervised learning. Since the network training is towards the direction of the most variant PCs, the network requires less training time to reach the optimal solution. We experimented with this hybrid dimension reduction on a collection of 350 images. The result is that it achieves the same accuracy level as the standard neural network approach in much less network training time.

Finally, we demonstrated the output quality of our hybrid method for indexing the test image collection using R^+tree. This shows that our proposed hybrid dimension reduction of image features can correctly and efficiently reduce the dimensions of image features and integrate the knowledge of human visual perceptions to the weights of the network. This enables any existing SAM to be used efficiently.

7.2 Future Work

One fruitful extension to this research will be combining the visual and topological features of images to form an unified index. The spatial feature provide an extra dimension to image similarity retrieval. As soon as these features are extracted and represented by digital values, they can be treated as the additional feature that is used to train the dimension reducer. Thus the similarity of images based on visual and topological features can become the measurement of the distance of image points in this unique feature space.

A current problem with our approach is the selection of network training data set. There are no formal methods to select the training data and new network architectures are needed so that the network has less dependence on the people who train the network. Optimal data analysis techniques may be employed to pre-select the network training data. It is worth studying a new network's architecture with the unsupervised learning algorithm to perform the nonlinear dimension reduction automatically without selecting the training samples manually.

Another research direction is to apply this approach to document retrieval. Document retrieval involves multiple attributes indexing. Our dimension reduction method can be extended to reduce the number of attributes that are needed to support an efficient multiple attributes indexing for fast document retrieval.

A final research issue is the study of the benefits of our hybrid dimension reduction for interactive multimedia data mining and visualization in multimedia databases. Multimedia data are high dimensional data that are difficult to be perceived by human visual systems. To manipulate these multimedia data interactively, it is necessary to reduce the dimensions of these data into a 2 or 3 dimensional space.

References

Anderberg, M.R. (1973), Cluster Analysis for Applications, New York: Academic Press, Inc.

Andreadis, I.; Browne, M.A. & Swift, J.A. (1990): Image pixel classification by chromaticity analysis. Pattern Recognition Letters 11 (1990) 51-58.

Ashley, J.; Barber, R.; W.; Flickner, M.; Hafner, J.;Lee, D.; Niblack, W.; Petkovic, D.;(1995): Automatic and Semi-Automatic Methods for Image Annotation and Retrieval in QBIC. *Proceedings SPIE Storage and Retrieval for image and Video Database III,* pp. 24 - 35.

Balakrishnan, P.V., Cooper, M.C., Jacob, V.S., and Lewis, P.A. (1994) "A study of the classification capabilities of neural networks using unsupervised learning: A comparison with k-means clustering", Psychometrika, 59, 509-525.

Baldi, P.; and Hornik, K. (89): Neural network and principal component analysis: learning from example without local minima. Neural Networks, Vol(2), pp. 53 - 58.

Bayer, R.; McCreigh, E.(72): Organization and maintenance of large ordered indices. *Acta Information,* Vol(1)3, pp. 173-189.

Bentely, J. L. (75): Multidimensional binary search trees used for associative searching. *ACM Communction*, vol(18)9, pp.509-517.

Bloomer, C.,M. (1989): Principles of visual Perception. Design Press.

Bourland, H, and Kamp, y. (88): Auto-Association by Multilayer Perceptrons and Singular Value Decomposition. *Biological Cybernetics.* Vol(59), pp. 291-294.

References

Chambers, J. M. (1977): Computational Methods for Data Analysis. New York: Wiley. pp. 111.

Chandrasekaran, S.; Manjunath, B. S.; Wang, Y. F.; Winkeler, J.; and Zhang, H. (1996): An Eigenspace Update Algorithm for Image Analysis. Techreport ECE, University of California, USA.

Comer, D.(79): The ubiquitous B-Tree. *ACM Computer Surveys.* Vol(11)2, pp. 121-137.

Daugman, J (85):Uncertainty relation for relation in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *J Opt Soc Am A 2:* pp.1160-1169.

Daugman, J. G.(85): Uncertainly relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. In J. Opt. Soc. Am. A, vol(2), pp. 1160 - 1169.

Deco, G. and Obradovic, D. (1996), An Information-Theoretic Approach to Neural Computing, NY: Springer-Verlag.

Duda, R. O.; Hart, P. E. (1973): Pattern Classification and Scene Analysis. Wiley, New York.

Euripides, G. M. Petrakis and Faloutsos, C (89): similarity Searching in Large Image Databases. pp.1-33.

Fahlman, S. E. (88): An empirical study of learning speed for Back-propagation networks. Technical report CMU-CS 88-162, Carnegie-Mellon University.

References

Faloutsos, C.; Barber, R.; Flickner, M.; Niblack, W.; Peetkovic, D. And Equitz,W (1994): Efficient and Effective Querying by image Content. *Journal of intelligent Information System*, pp. 231-262.

Finkel, R. A.; and Bently, J. L.(74): Quad Trees: A data structure for retrieval on composite keys. *Acta informatical 4*, pp. 1-9.

Flickner, M.; Sawhney, H.; Niblack, W.; Ashley, J.; Huang Q.; Dom, B.;Gorkani,M.; Hafner, J.; Lee, D.; Peetkovic, D.; Steels,D and Yanker Peter. (1994): Query by Image and Video Content The QBIC System. *IEEE computer*, September.

Fukunaga, K and Koontz, W (70): Application of the Karhunen-loeve expansion to feature selection and ordering. *IEEE transactions*

Gabor, D (93): Theory of communication: *JIEE(London)*, pp.429-457.

Garrison, W. Cottrell and Janet Metcalfe. (91): EMPATH: Face, emotion, and gender recognition using holons. *Advances in neural Information Processing System 3*, Morgan Kauffmann, pp. 564-571.

Girshick, M. A. (1939): On the sampling theory of roots of determinant equations

Gnanadesikan, R. (1977): Methods for Statistical Data Analysis of Multivariate Observations, New York: Wiley. pp. 10.

Golomb, B. A.; Lawrence, D. T.; and Sejnowski, T. J.: Sexnet: A neural network indentifies sex from human faces. *Advances in neural Information Processing System 3,* Morgan Kauffmann, pp. 572-577.

Gonzalez, R. C. and Woods, R. E. (1992): *Digital Image Processing*. *Addison-Wesley*, pp. 508-510.

References

Gonzalez, R. C. and Woods, R. E.: Digital Image Processing. Addision-Wesley, pp. 508-510, 1992.

Gudivada, Venkat N and Raghavan, Vijay V (1995): Content-Based Image Retrieval Systems. *IEEE computer*, September, pp.18 - 22.

Gunther, O. (88): Efficient structure for geometric data management. Lecture Notes in computer Science 337, Springer-Verlag

Guttman, A. (84): R-tree: A dynamic index structure for spatial searching. *Proceeding of ACM SIGMOD International Conference on Management of Data.* PP. 47-57.

Haralick, R, M (79): Statistical and structural approaches to texture. *Proc. IEEE, 67,* PP.786-804.

Hermansen, F.; Bloomfield, P. M.; Ashburner, J. (95): Linear dimension reduction of sequences of medical images: II. Direct sum decomposition. *Physic in medical and biological.* Vol (40), pp. 1921-1941.

Hertz, J., Krogh, A., and Palmer, R. (1991). Introduction to the Theory of Neural Computation. Addison-Wesley: Redwood City, California.

Hinrichs, K.; and Nievergelt, J. (1983): The grid file: A data structure designed to support proximity queries on spatial objects. *Proceeding of International Workshop on Graphtheoritic Concepts in Computer Science.* Pp. 100-113

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. J. Educ. Psychol., 24, 417-441, 498-520.

Huang, W.Y.; Lippmann, R.R. (1988): Neural nets and traditional classifier. *Neural Information Procession System (Denver, 1987).* D. Z. Anderson, American Institue of Physics, New York. Pp. 387-296.

References

Hush, D. R; Salas, J. M.; and Horne, B. (1991): Error surface for multi-layer perceptrons. *International Join Conference on Neural Network.* Vol (I), pp. 759-764, IEEE, New York.

Jackson, J.E. (1991), A User's Guide to Principal Components, NY: Wiley.

Jolliffe, I.T. (1986), Principal Component Analysis, Springer-Verlag.

Joseph, T.; and Cardenas, A. F. (1988): PICQUERY: a high level query language for pictorial database management. *IEEE Transaction on Software Engineering.* Vol(14) SE.

Karhunen, J. (1994), "Stability of Oja's PCA subspace rule," *Neural Computation*, 6, 739-747.

Kittler, J and Young, P. (1973): A new application to feature selection based on the Karhunen-loeve expansion. *Pattern recognition*, Vol (5), pp. 335-352.

Knuth, D. E.(1973): Fundamental Algorithms, 2nd Edition. *The art of computer programming.* Vol (1), Reading, MA, Addison-Wesley.

Kohonen, T. (1995), Self-Organizing Maps, Berlin: Springer-Verlag.

Kohonen, T. (1984): Self-Organization and Associative Memory. Springer-Verlag, Berlin.

Kosko, B. (1992), Neural Networks and Fuzzy Systems, Englewood Cliffs, N.J.: Prentice-Hall.

Kramer, M. A.; (1991): Nonlinear proinciple component analysis suing auto-associative neural networks. *AIChE Journal*. Vol(37), pp. 233-2243.

References

Lee, D.; Barber, R.; W.; Niblack, W.; Flickner, M.; Hafner, J.; Petkovic, D.; (1993): Indexing for Complex Queries on a Query-By-Content Image. *Proceedings SPIE Storage and Retrieval for image and Video Database III*, pp. 24 - 35.

Leung, C.H.C; Hibler, J and Mwara, N (1994): Content-based Retrieval in Multimedia Databases. *Computer Graphics*, VOL 28(1) 24-28.

Ljubomir J. Buturovic (1993): Toward Bayes-Optimal Linear Dimension Reduction. *IEEE Transaction on Pattern Analysis and Machine Intelligence.* Vol(14)4, pp. 420-424.

Lu, H.; Ooi, B.C &Tan, K. L.: Efficient Image Retrieval by Colour Contents. *Department of Information System and Computer Science*, National University of Singapore.

MacQueen, J.B. (1967), "Some Methods for Classification and Analysis of Multivariate Observations," *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1, 281-297.

Mandel, J. (1982): Use of the singular value decomposition in regression analysis. Amer. *Statistician*, 36, pp.15-24.

Markus, H. Grob. And Seiber, F. (1993): Visualization of multidimensional image data sets using a neural network. *The visual Computer.* Vol(10), pp. 145-159.

Mehrotra, R.; and Gray, J. E.: Similar-Shape Retrieval in Shape Data Management. *IEEE Computer, September,* pp. 12-24.

Mulier, F. and Cherkassky, V. (1995), "Self-Organization as an Iterative Kernel Smoothing Process," *Neural Computation*, 7, 1165-1177.

Namphol, a.; Arozullah, M; and Chin, S. (1991): Higher order data compression with neural networks. *Proceedings of the IJCNN*, pp. 55 - 59.

References

Nelder, J. A. (1985): An alternative interpretation of the singular value decomposition in regresion. Amer. *Statistician*, 33, pp.313-315.

Niblack, W.; Barber, R.; Equitz, W.; Flickner, M.; Glasman, E.; Petkovic, D.; Yanker, P.; Faloutsos, C. & Taubin, G. (1993): The QBIC Project: Querying Image by Content Using Colour, Texture and Shape". *Proc. of SPIE* 1908, pp. 173-178.

Ogle, V. E.; and Stonebraker, M. (1995): Chabot: Retrieval from a Relation Database of image, *IEEE Computer*, September, pp. 40 - 48.

Oja, E. (1991): Data compression, feature extract, and auto-association in feedforward neural networks. *Artificial Neural Networks.* Elsevier Science Publishers, North-Holland, pp. 737-745. *on computer,* Vol(c)9, pp. 311-318.

Ooi, B. C. (1989): Efficient Query Processing in a Geographic Information System. Ph.D. Thesis, Monash University.

Orenstein, J.; and Manola, F. (1991): probe: spatial data modeling and query processing in an image database application. *IEEE Transaction on Software Engineering.* Vol(14) SE.

Orenstein, J. (1986): Spatial Query Processing in an Object-Oriented Database System. *Proceeding of ACM SIGMOD.* PP. 326-336.

Pearson, k. (1901): On linew and planes of closet fit to system of points in space.Phil. Mag. (6), 2, 559-572.

Pentland, A.; Picard, R.; and Sclaroff, S. (1994): Photobook: Tools for Content-Based Manipulation of image databases. *SPIE Proceeding,* Vol (2)185, pp. 34-37.

Pickard, R. W.; and Minka, T. P. (1995): Vision Texture for Annotation. *Multimedia System,* Vol (3), pp. 3 - 14.

References

Riedmiller, M. (1994): Advanced supervised learning in multi-layer perceptrons from backprogation to adaptive learning algorithms. *Computer Standards & Interface* Vol(16) pp.265-278.

Robinson, J. T. (1981): A search structure for large multimedimensional dynamic indexes. *Proc. ACM SIGMOD International conf. On Management of Data, Ann Arbor, Michigan*, pp. 10-18.

Roussopoulos, N.; and Leifker, D. (1985): Direct spatial search on pictorial databases using packed R-tree. *Proceeding of ACM SIGMOD International Conference on Management of Data*, pp. 17-31.

Rumelhard, D.; Hinton, G.; and Williams, R. (1986): Learning internal representations by error propagation. *Parallel Distributed Processing.* Mit Press, Cambridge, MA pp. 318-362.

Samet, H. (1989): The Design and Analysis of Spatial Data Structures. *Addison-Wesley.*

Saund, E (1980): Dimensionality-Reduction Using Connectionist Networks. *IEEE transaction on pattern analysis and machine intelligence,* Vol(11)3, March, pp.304-314.

Seeger, B.; and Kriegel, H. (1988): Techniques for design and implementation of efficient spatial access methods. *Proceeding of Fourteenth International Conference on Very Large Database.* pp. 360-371.

Sekuler, R. and Blake, R. (1994): PERCEPTION. McGraw-Hill, INC.

Sellis, T.; Roussopoulos, N.; and Faloutsos, C. (1987): The R^+-tree: a dynamic index for multimensional objects. *Proc. VLDB, Brighton*.

Shao, J and Forstner, W (1994):Gabor wavelets for texture edge extraction. *Comm. II symposium(WG/2)*. PP. 745-752.

References

Sirovich, L.; Kirby,M. (1987): Low-dimensional produce for the characterization of human faces. J Opt Soc Am[a]4(3): pp. 519-206.

Stricker, A, Markus (1994): Bounds for the discrimination power of color indexing techniques. *Proceedings SPIE Storage and Retrieval for image and Video Database II,* pp. 15-24.

Swain, M.J. & Ballard, D.H. (1991): Color Indexing. International Journal of Computer Vision, Vol (1). Pp.11-32.

Tam,S.W.S (1995): Image Retrieval in Multimedia System. Honor Thesis Report, The University of New South Wales.

Tarassenko, L. and Roberts, S. (1994), "Supervised and unsupervised learning in adial basis function classifiers," *IEE Proceedings-- Vis. Image Signal Processing*, 141, 210-216.

Tiu, L, S (1995): Image Retrieval in Multimedia System. Honor Thesis Report, The University of New South Wales.

Usui, S.; Nakauchi, S.; and Nakano, M. (1991): Intenal colour representation acquired by a five-layer neural network. *Artificial Neural Networks.* Elsevier Science Publishers, North-Holland, pp. 100-120.

Watanabe, S (1965): Karhunen-loeve expansion and facto analysis. *Transactions 4th prague Conference of Information and theory.*

Watanabe, S (1970): Feature compression. *Advance in information System Science,* Plenum, Vol(3), pp. 63-111

Wilking, J. H. (1965): The Algebraic Eigenvalue Problem. Oxford: Oxford University Press.

References

Wu, J. K.; Narasimhalu, D. A.; Mehtre, B. M.; Lam, C. P.; Gao, Y. J. (1995): CORE: A Conten-Based Retrieval Engine for Multimedia Information Systems. *Multimedia System*, Vol(3), pp. 25 - 41.

Zhang, H. J.; and Zhong, D. (1995): A Scheme for Visual Feature Based Image Indexing. *Proceedings SPIE Storage and Retrieval for image and Video Database III,* pp. 36-46.

Appendix 1

Sample of images with colour and texture feature vector

Image with their corresponding 1-D colour feature

image 1

H=0.655555

image 2

H=0.644344

image 3

H=0.644444

image 4

H = 0.5916667

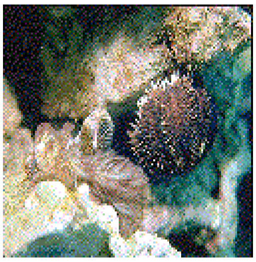


image 5

H = 0.5972222

image 6

H = 0.583337

image 7

H = 0.333333

image 8

H = 0.333222

image 9

H = 0.208333

Appendix 1

Image and their 30-D texture feature vectors

image1	image2	Image3	Image4	Image5	Image6	Image7	Image8	Image9
0.249790	0.220799	0.351102	0.389428	0.235917	0.250624	0.349011	0.222217	0.266565
0.172207	0.143780	0.181570	0.176020	0.332536	0.138263	0.133491	0.131483	0.092028
0.304605	0.196199	0.222239	0.254597	0.404272	0.285163	0.220959	0.224853	0.280858
0.217306	0.235551	0.264908	0.254597	0.404272	0.285163	0.220959	0.205918	0.280858
0.211042	0.192940	0.249619	0.464932	0.200069	0.317059	0.201017	0.190556	0.274055
0.356250	0.282070	0.183500	0.235768	0.438867	0.666295	0.301692	0.275183	0.271719
0.188145	0.169757	0.213822	0.200132	0.230152	0.314305	0.160160	0.354979	0.149340
0.210169	0.149198	0.199251	0.280644	0.225883	0.290796	0.262152	0.132836	0.113410
0.210169	0.326972	0.233727	0.485323	0.330709	0.458635	0.262152	0.150840	0.238523
0.271060	0.185988	0.249947	0.215855	0.272261	0.315452	0.191037	0.321931	0.158969
0.174274	0.186785	0.141294	0.261892	0.226867	0.346267	0.126112	0.218832	0.184578
0.122227	0.095400	0.101253	0.157230	0.081871	0.164497	0.098148	0.237959	0.073653
0.080140	0.087992	0.093741	0.173603	0.130983	0.191302	0.098683	0.137386	0.075826
0.076936	0.087696	0.080426	0.192641	0.113662	0.209020	0.098683	0.140773	0.177822
0.110793	0.086879	0.111116	0.117873	0.100312	0.171523	0.078796	0.264512	0.055245
0.238924	0.331671	0.288466	0.225708	0.365813	0.555075	0.445851	0.377296	0.325769
0.128925	0.129583	0.146700	0.179484	0.173825	0.240650	0.156890	0.272807	0.136449
0.174234	0.156331	0.128610	0.207756	0.193914	0.343260	0.214416	0.248915	0.182988
0.153575	0.156331	0.128610	0.208244	0.193914	0.343260	0.166791	0.291345	0.201191
0.113756	0.116115	0.123394	0.221768	0.180180	0.304866	0.181968	0.344673	0.128744
0.363120	0.293045	0.167410	0.320730	0.275774	0.320282	0.433671	0.524885	0.294543
0.194136	0.157396	0.168409	0.270094	0.248582	0.197117	0.290079	0.357646	0.195805
0.197629	0.156883	0.167860	0.257957	0.225415	0.315276	0.255004	0.283423	0.185837
0.184591	0.156883	0.167860	0.271682	0.206686	0.260468	0.297633	0.402318	0.245687
0.166049	0.157396	0.168409	0.214736	0.214781	0.215322	0.235467	0.377970	0.193962
0.859873	0.815065	0.872094	0.906352	0.870226	0.848111	0.800846	0.805360	0.804868
0.033129	0.031403	0.033600	0.034920	0.033528	0.032676	0.030855	0.031029	0.031010
0.338488	0.320850	0.343299	0.356785	0.342564	0.333858	0.315252	0.317029	0.316836
0.338488	0.320850	0.343299	0.356785	0.342564	0.333858	0.315252	0.317029	0.316836
0.033129	0.031403	0.033600	0.034920	0.033528	0.032676	0.030855	0.031029	0.031010

Appendix 1