IMAGE CLASSIFICATION FOR CBIR USING NEURAL NETWORK

By

Kum. Manju

07MCE007

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

May 2009

IMAGE CLASSIFICATION FOR CBIR USING NEURAL NETWORK

Major Project

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Computer Science and Engineering

By

Kum. Manju

07MCE007

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

May 2009

Certificate

This is to certify that the Major Project entitled "Image Classification for CBIR using Neural Network" submitted by Kum. Manju (07MCE007), towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science and Engineering of Nirma University of Science and Technology, Ahmedabad is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, havent been submitted to any other university or institution for award of any degree or diploma.

Prof. Swati Jain
Guide, Assistant Professor,
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Dr. S. N. Pradhan
Prof. P.G. coordinator,
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Prof. D. J. Patel
Professor and Head,
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Dr. K. Kotecha
Director,
Institute of Technology,
Nirma University, Ahmedabad

Abstract

The recent development of computing hardware has resulted in a rapid increase of visual information such as databases of images. To successfully utilize this increasing amount of data, we need effective ways to process it. Content-based image retrieval utilizes the visual content of images directly in the process of retrieving relevant images from a database. The retrieval is based on visual features such as the colors, textures, shapes, and spatial relations the image contains rather than traditional textual keywords.

This work is to cluster similar images in a large, unannotated image database. In which for each image a set of average RGB color is calculated and a SOM is trained for each feature.After that images are clustered in different classes.

Results have shown that they match better with the human perception. MATLAB 7 Is the tool used to implement image processing algorithms and Microsoft Access DBMS used for managing the database of image features vector.

Keywords:

Image classification Content-based image retrieval, image databases, self-organizing map.

Acknowledgements

It gives me great pleasure in expressing thanks and profound gratitude to Dr. S. N. Pradhan, P.G. coordinator, Department of Computer Engineering, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout the Major Project. I heartily thankful to him for his time to time suggestion and the clarity of the concepts of the topic that helped me a lot during this study.

I would like to give my special thanks to Prof. Swati Jain, Department of Computer Engineering, Institute of Technology, Nirma University for guiding throughout the Major Project and providing feedback to improve my work.I would like to give my thanks to Prof. D.J .Patel, Head, Department of Computer Engineering, Institute of Technology, Nirma University for his continual kind words of encouragement and motivation throughout the project.

I am thankful to all faculty members for their special attention and suggestion towards the seminar work. I extend my sincere thanks to my colleagues for their support in my work. I am really thankful to GOD who gives me courage to face difficulties and overcome them.

Last, but not the least, no words are enough to acknowledge constant support and sacrifices of my family members because of whom I am able to complete the degree program successfully.

Kum. Manju

(07MCE007)

Contents

List of Tables

List of Figures

List of Abbreviation

2D	Two Dimension
3D	Three Dimension
ANN	Artificial Neural Network
BMU	Best Matching Unit
CIS	Laboratory of Computer and Information Science
CBIR	Content-Based Image Retrieval
CBIQ	Content-Based Image Query
DBMS	Data Base Management System
GUI	Graphical User Interface
HSV	Hue, Saturation, Value
kNN	k-Nearest-Neighbor
MPEG	Moving Picture Experts Group
QBE, QBPE	Query by (pictorial) example
RGB	Red Green Blue
SOM	Self-Organizing Map
TS-SOM	Tree Structured Self-Organizing Map
UI	User Interface
VQ	Vector Quantization
WWW	World Wide Web
PCA	Principal Component Analysis
SVD	Singular Value Decomposition
MDS	Multi Dimensional Scaling

Chapter 1

Introduction

1.1 General

Thousands of images are generated every day, which implies the necessity to classify, organize and access them by an easy and faster way. Image categorization is an open problem in machine vision and has many attractive applications, such as image annotation, retrieval and scene understanding. The existing approaches are based on a wide range of technologies, ranging from the design of the features to the classification schemes.

These approaches follow a similar principle: training an image classifier using low level visual cues such as color, shape, texture and etc., as features. Digital image and video libraries are becoming more common and widely used as more visual information is produced at a rapidly growing rate. Creating and storing digital images is nowadays easy and it is getting cheaper all the time as the needed technologies are becoming available to the masses. There already exist a vast number of digital visual data sources, e.g. different kinds of sensors, digital cameras and scanners in addition to various image collections and databases for all kinds of purposes. Furthermore, the fast development of computing hardware has enabled us to switch from text-based computing to graphical user interfaces (GUIs) and multimedia applications. This

transition has fundamentally changed the use of computers and made visual information an inseparable part of everyday computing.

As a result of all this, the amount of information in visual form is increasing and we need effective ways to process it. The existing and widely adopted methods for text-based data are usually inadequate for these purposes. Visual information is, generally, said to include both still images and moving video and sometimes even three-dimensional environments built with virtual reality models.

Traditional text-based image retrieval systems use keyword annotations as the retrieval paradigm. This approach does, nevertheless, have some obvious shortcomings and difficulties. Annotating large databases takes a lot of effort as the annotations must be entered manually. Another problem is the possibility of different interpretations of the image content. Different people see different things in images and the annotations cannot possibly cover them all. As a result, a fraction of potentially relevant images may not be included in the result of a query.

Content-based image retrieval (CBIR) is another approach to the problem. It is based on automatically extracted features from the contents of the image.Popular features used in most current applications include the colors and different textures in the image, the shapes of the objects represented, and the structure of the image.This approach does also have some unavoidable difficulties. Humans also have subjective opinions and different people see and highlight different aspects in images.

1.2 Image classification

Classification is a process that groups data in categories possessing similar characteristics. Clustering algorithms are useful for high-dimensional data where it is impossible for us to visualize the data in space. A clustering mechanism groups similar image feature vectors provided the feature vectors are extracted in a meaningful way.

For image categorization or classification, it is assumed that a fixed set of classes or categories has been defined, and that each image belongs to one category. In a

typical scenario in an image classification problem, we have a set of unlabeled images, and we want to assign each of these images into one of the known or unknown categories. For an automatic approach to this problem to be feasible, one needs to assume that images that are in a way similar will fall into the same category, and that this underlying notion of similarity can be captured automatically using a suitable representation of images and some learning algorithm.

Image retrieval, where the notions of representation and similarity are of great importance, is a sub-problem of image categorization. One way of applying results from image retrieval to the problem of image categorization would be to take simple vector based descriptions of images, such as histograms or autocorrelograms, and use any of the numerous ML techniques that can work with vectors.

A common approach to image classification involves addressing the following three issues: (i) how to represent an image, (ii) how to organize the data, and (iii) how to classify an image. Acquiring nice features and carefully modeling, the feature data are vital steps in this approach. Common features include color, texture, and shape information of an image. Some also integrate visual information and text accompanying an image.

Classification is a process that groups data in categories possessing similar characteristics. Clustering algorithms are useful for high-dimensional data where it is impossible for us to visualize the data in space. A clustering mechanism groups similar image feature vectors provided the feature vectors are extracted in a meaningful way.

1.3 Objective of Study

The main motivating factors for selection of this topic for dissertation are :

- Main goal will be to develop an image classification system reducing the amount of manual supervision required as well as reducing the computational cost to learn the classifier.

- Given a set of images and main objective is to extract feature from each image and classify it.

- There are lots of categories used so it is not enough to use only one feature (say the shape) of the objects to distinguish amongst them. For example shape may be a good feature to distinguish between cars and airplanes but it is not good to distinguish between horses and zebras.

- Main work is to use features and combination of features which provide a discriminative image representation amongst all the categories.

1.4 Motivation

There are lots of applications that can benefit from the image classification:

- **Image search.** Image search is the most direct application when people talk about image classification. In this sense, we can think about searching images in the biggest database in the world, Internet image search engines, or simply provide applications to search images in a personal computer.
  Nowadays the poor performance when searching images in Internet is due to their use of the image filename or surrounding HTML rather than the actual image content. However, the natural way to find images is to search visually -as humans due- using computer vision methods. Moreover, many companies have large archives of images which they wish to search in.

- **Video search.** Lots of adverts and video data have been generated during last years. People working in marketing are often interested in look for coffee adverts televised in the past years, or adverts filmed in the mountains. Nowadays all these adverts are manually annotated and stored in databases using meta data information. It would be very useful to provide techniques to access them automatically, by its content. Also producers or film directors would be

interested in recover by an automatic way those shots of movies filmed near a lake, or those shots where Johnny Deep appears in the middle of the ocean.

- **Medical applications.** In the medical field also lots of images are generated every day, radiographies, geographies etc. It would be very useful for the doctors to pro-vide tools to access at these images faster and not looking case by case as they do. Even though one can thing that this field is very different from the objective of the thesis we will see in Appendix B that there is a lot in common.

- **Travel guide.** With the current spread of cheap flights people travel every day more and more. Instead of having a travel guide of each country we can have a digital travel guide stored in the mobile phone and retrieve the information by taking a picture from the famous cathedral, square.

- **Video Compression.** Due to the very limited bandwidth of a number of important communication channels (e.g. wireless, underwater, low-power camera networks, etc.), video communication over such channels requires substantial compression of the video signal. One of the most promising answers to this challenge is to adopt a new compression paradigm that relies heavily in scene understanding. It would allow the compression of different objects in a scene with specific compression levels in such a way as to adjust the trade-off between space reduction and visual quality on a per-object basis. The basic idea is that important objects such as actors should retain the highest visual quality, while objects in the background can be encoded with lower quality to save bytes. Here, computer vision must help to perform automatically the task of separating a video into the objects of which it is composed.

- **Surveillance.** Fundamental systems remain relatively unintelligent requiring a per-son screening the image sequences, looking for suspicious people and unusual events. Advanced systems try to automatically detect this unusual event. A subject of relative importance is that related to understand crowded environ-

ments (e.g. a football stadium) detecting risk situations (e.g. fights).

- **Aerial images.** National mapping agencies spend thousands of euros each year to keep their data up to date. This tedious and time consuming process often involves a person in front of a computer screen comparing the current raster/vector map with the most recent high quality satellite image. Image classification techniques could be used to detect landscape changes with minimal human interaction.

- **Robotics.** Provide an eye to a robot is maybe one of the most ambitious things in the computer vision field. In this way a completely autonomous robot specialized to recognize certain objects of interest will be able to substitute humans in dangerous situations such as underwater exploration, fireman help etc.

1.5 Scope of Work

Classification is a process that groups data in categories possessing similar characteristics.The term image categorization refers to the label of images into one of a number of predefined categories. Clustering algorithms are useful for high-dimensional data where it is impossible for us to visualize the data in space. A clustering mechanism groups similar image feature vectors provided the feature vectors are extracted in a meaningful way.
The work of dissertation is limited to labeled the group of classes in particular category.Use of self-organizing map toolbox helps me training the SOM network by using average RGB color feature and classify images in different categories.

1.6 Thesis Organization

The following chapters of this thesis are organized as follows.

Chapter 2 describes the existing systems and the literature survey done. It gives a brief description of the related work.

Chapter 3 discusses extracting various statistical features from images to represent and compare image content. The main feature types described are color, texture, shape, and structure of the images as they are by far the most common features in current CBIR applications.

Chapter 4 describes Artificial Neural Networks in Classification Problems, in this Kohenen Neural Network is used for classified the images.The Self-Organizing Map (SOM)[1] is an unsupervised, self-organizing neural algorithm widely used to visualize and interpret large high-dimensional data sets.

Chapter 5 describes Implementation and Results, For each image a set of Average RGB color is calculated and a SOM is trained for each feature.SOM is used for similarity based clustering.By using SOM toolbox [2] database images classified in different-different classes.

Finally, the conclusions of this thesis and directions for future research are discussed in **Chapter 6**.

Chapter 2

Literature Review

.

2.1 General

The Self-Organising Map is a neurally-motivated unsupervised learning technique which has been used in many data analysis tasks. A genuine feature of the Self-Organizing Map is its ability to form a nonlinear mapping of a high-dimensional input space to a typically two-dimensional grid of artificial neural units. During the training phase of a SOM, the weight vectors in its neurons get values which form a topographic or topology-preserving mapping. In which vectors that reside near each other in the input space are mapped in nearby map units in the output layer. Patterns that are mutually similar in respect to the given feature extraction scheme are thus located near each other on the SOM.

2.2 Literature review

PicSOM uses the Tree Structured Self Organizing Map (TS-SOM) as the method for scoring image similarities. The object has been to utilize the strong self-organizing power of the Self-Organizing Map (SOM) in unsupervised statistical data analysis

for images. The approach is believed to facilitate content-based retrieval from image databases ranging from small and domain-specific picture sets to large and unstructured collections of miscellaneous images.

2.3 Basic Operation of PicSOM

The implementation of PicSOM is based on a general framework in which the interfaces of co-operating modules are defined. Consequently, the current components are easy to replace and multiple modules can also be used in parallel. Even the Tree Structured Self-Organizing Map is only one choice for the similarity measure. Nevertheless, the results gained so far are promising on the potentials of the current TS-SOM method.

PicSOM may utilize one or several types of statistical features describing the color, texture, shape, and structure of the images. A separate TS-SOM is trained for each feature set and the maps are used in parallel to resolve the best-matching images. After the training phase, the map units are labeled with images belonging to the database.

For each map unit of the TS-SOMs, the feature vector which is closest to the stored model vector of the map unit is determined. The corresponding image is then used as a representative image or label for that particular map unit. As a result, a tree-structured hierarchical representation of all the images in the database is formed. This is illustrated in Figure 2.1 in which two consecutive map levels with sizes of $4*4$ and $16*16$ are displayed. The map was trained with the YIQ color feature using the ftp.sunet.se database. The intrinsic property of the SOM that it organizes similar models close to each other can be particularly observed from the lower map as images with similar color content are located near each other. A novel technique introduced in the PicSOM system facilitates automatic combination of the responses from multiple TS-SOMs and all their hierarchical levels. This mechanism aims at autonomous adaptation to the user's behavior in selecting which images resemble each other in

Figure 2.1: The surfaces of the first two map layers (sized 4∗4 and 16∗16) of the ftp.sunet.se database based on the YIQ color feature.

the particular sense in which the user seems to be interested. During the retrieval process, PicSOM then tries to adapt to the user's preferences regarding the similarity of images. This method can thus be seen as a SOM-based approach to relevance feedback. Another intrinsic feature in the system is its ability to use multiple reference images. This feature makes PicSOM differ from other content-based image retrieval systems, such as QBIC , which use only one reference image at a time.

With the current PicSOM user interface, the image queries are performed through a standard World Wide Web (WWW) browser. This makes image querying with PicSOM machine independent and makes it possible to provide the system in the WWW.

The journal articles and conference papers listed below are included in this thesis. In this section, the content of each paper is briefly described:

I. Jorma Laaksonen, Markus Koskela, Sami Laakso, and Erkki Oja (2000)[3]. PicSOM - "**Content-Based Image Retrieval with Self-Organizing Maps**": This is the first journal article on the PicSOM system. It contains a concise description of all the major parts of the system. The visual features used in early experiments in the system are described.

II. Jorma Laaksonen, Erkki Oja, Markus Koskela, and Sami Brandt (2000)[4]. "**Analyzing Low-Level Visual Features using Content-Based Image Retrieval**": This paper discusses the analysis of low-level statistical visual features in a CBIR setting. CBIR is seen as an emerging research topic benefiting from previous research on natural image statistics. The relevance of different statistical features can be evaluated in the PicSOM setting. While the connection from low-level features to semantic concepts remains unsolved, the use of a meaningful set of parallel features can aid in linking statistical visual features to image similarity perceived by humen. The illustrations of semantic image classes on feature-wise SOM surfaces are also introduced.

III. Jorma Laaksonen, Markus Koskela, Sami Laakso, and Erkki Oja (2001). "**Self-Organizing Maps as a Relevance Feedback Technique in Content-Based Image Retrieval**": This is a journal article focusing on the description and qual-

itative analysis of the relevance feedback technique based on Self-Organizing Maps which is the backbone of the PicSOM system. This article also covers other existing relevance feedback methods and different kinds of usage of SOMs in the CBIR field. Advantages and differences of our method when compared to existing relevance feedback methods are discussed. A general CBIR system structure and the idea of dividing the task of a CBIR system into independent stages or blocks is introduced by separating the per-feature and final processing stages, resulting in reduced computational burden. Initial version of the reference methods based on vector and scalar quantization are introduced.

IV. Markus Koskela, Jorma Laaksonen, and Erkki Oja (2001). "**Comparison of Techniques for Content-Based Image Retrieval**": It takes the block structure approach further. The operation of a CBIR system is seen as a series of independent processing stages. For each stage, there may exist multiple choices, and different CBIR systems may be implemented in this framework. Moreover, each stage of processing may be analyzed separately. Performed experiments validate these assumptions and show two alternative paths in the block structure that lead to superior results.

V. Jorma Laaksonen, Markus Koskela, and Erkki Oja (2002). PicSOM-"**Self- Organizing Image Retrieval with MPEG-7 Content Descriptions**": This is a journal article describing an extensive evaluation of the PicSOM system with visual content descriptors defined in the MPEG-7 standard. In this work, previous features have replaced with ones defined in MPEG 7. In addition, a slightly modified version of algorithm was presented and used. The results were presented using recall-precision curves and they show that PicSOM can readily utilize the MPEG-7 content descriptors and the system in general benefits from using as many descriptors as there are available without any preceding feature selection.

VI. Markus Koskela, Jorma Laaksonen, and Erkki Oja (2002). "**Implementing Relevance Feedback as Convolutions of Local Neighborhoods on Self-Organizing Maps**": It discusses the interpretation of PicSOM's relevance feedback technique as convolutions of sparse value fields obtained from the relevance information with a

kernel function. A number of kernel functions with different sizes are compared in spreading the relevance information on the SOM surfaces. In addition, two methods for incorporating information about the relative distances of the map units in the original feature space are presented.

VII. Markus Koskela and Jorma Laaksonen (2003). Using Long-Term "**Learning to Improve Efficiency of Content-Based Image Retrieval**": It presents a method to use previously recorded user-system interaction data as an image feature which can be used to improve retrieval efficiency on large databases of miscellaneous images. The method can also be used for existing keyword annotations, which can result in greatly improved retrieval results.

Chapter 3

Feature extraction and indexing

Image features refer to characteristics which describe the contents of an image. In a broad sense, these include visual features extracted directly from the image, textual keywords, and miscellaneous image meta-data, such as the file name, image format and size. Various types of image features are discussed in depth in this chapter.
In this thesis, the word feature is used to refer both to a single scalar value describing a certain characteristic of an image as well as whole sets of these values which constitute fully-functional image representations, e.g. the RGB color feature. Depending on context, the intended meaning of the word should be clear. Moreover, a feature vector is a compilation of related scalar values, gathered into vector form.

3.1 Overview

Visual feature extraction is the foundation for all kinds of applications of content-based image retrieval and, therefore, various types of features have been studied extensively. Most of the early work in this area has been concentrated on finding the best viable features and indexing methods to represent the similarities between images. Even the Moving Picture Experts Group has started to work on a standard called MPEG 7 or Multimedia Content Description Interface, partly to develop a set of standard features for image content description.

Different visual features can be extracted either with automatic or semi-automatic methods. Fully-automatic feature extraction is obviously very appealing, especially with large image databases, as it requires a minimal amount of human effort. Current knowledge on image analysis and pattern recognition is still limited and hence the automatic methods cannot always provide sufficient discriminating power for effective image retrieval.Semi-automatic methods require some human assistance in tasks like image segmentation. This often improves the performance but, depending on the application, can require too much human effort to be feasible. For instance, as reliable shape recognition is a difficult task for a computer, manually pointed object contours may enhance shape detection substantially.

The size of the image database sets limits to the complexity of practical features. With excessively large databases, it is not feasible to extract computationally demanding features from each image. Another affecting factor is the average size of the images, as the computational requirements are usually a function of the number of pixels in the image. Overall, the feature vector computation should be as efficient as possible.

Usually, the general-purpose features, applicable for a variety of image types, are said to include color, texture, shape, and structure.

The representation of the content of an image I is usually compiled into a d-dimensional I feature vector f^I :

$$f^I = ({f_1}^I {f_2}^I {f_3}^I ... {f_d}^I)^T. \tag{3.1}$$

A typical feature vector dimension d in content-based image retrieval applications is of order 100.

The measurement of similarity or dissimilarity between two feature vectors is a fundamental task. One commonly used scalar measure of vector dissimilarity is distance. We can use some distance metric for the vectors f^I and f^J to measure the dissimilarity between two images I and J with respect to a given feature extraction method.

The most natural choice is the Euclidean or L_2 metric:

$$d_{L_2}(I,J) = \sqrt{(f^I - f^J)^T(f^I - f^J)} = \sqrt{\sum_{k=1}^{d}(f_K^I - f_K^J)^2}. \quad (3.2)$$

Other metrics, such as the L_1 and L_∞ metrics, are used in some applications. It is also possible to use weighted distance measures. The efficiency of the distance measurement is also of importance as it is generally made on-line.

An important part of designing a CBIR application is to select relevant features to characterize the images. A suitable feature should contain sufficient discriminating power to distinguish images representing different things. On the other hand, different images of the same object often differ from each other, for example, in position, angle, and scale of the object. A good feature should still categorize these images together, i.e. it should be invariant to such transformations.

If the values of f_i^I in 3.1 are unconstrained, the resulting feature space is $\Re^d$.And, as an example, should we use the pixel values of an image directly as features, a 256∗256-sized image I would be transformed into a feature vector f^I in a 65536-dimensional space. Processing large numbers of such vectors is clearly infeasible due to massive storage and computation requirements.Therefore,during the feature extraction process,the dimensionality of the data is reduced.Good features should still maintain those characteristics which increase the discriminating power of the feature while excluding any redundant information.

3.2 Color

Color is a simple and straightforward feature for all kinds of color images. The human eye is much more sensitive to color shades than gray-level intensities in an image. The colors of different objects are also resolution and view invariant. The color characteristics of an image are often an important attribute of the image content.

Some common materials and backgrounds have distinct color properties: the sky is blue, flora is green, human skin has a distinguishable color, and so on. Different objects in images are, however, not generally identified by their color attributes. Queries based on color may thus retrieve images with similar color distributions but which represent totally different objects. Furthermore, with color information it is possible to segment different regions from the image and use their color contents separately.

3.2.1 Methods for Color Feature Extraction

In content-based image retrieval, color has been the most commonly-used feature type.It is relatively easy to utilize and it usually yields reasonable results which can then be improved by adding other types of features. The main methods for representing color information are described in this section.

Color:AverageRGB

The average color of an image is a simple feature. Still, it can be useful with some types of images. The average values of the color bands are computed and used as features to retrieve similar images. It is also possible to compute the average values in several color spaces and use them together in the feature vector. If the images contain certain known objects and their positions are not of interest, the global average color feature may be able to retrieve the desired images with a tolerable accuracy.

The global color distribution has a limited discriminating power as it does not consider the layout of different colors in the images. Thus, with large databases it often produces a large number of false positives, i.e. images which happen to have similar overall color distributions but different contents.

The average values for red, green, and blue color channels ($\bar{r_j}$, $\bar{g_j}$, $\bar{b_j}$) are calculated in the five separate zones I_j, j = 0,1,..., 4 of the image as seen in Figure 3.1. The circular zone, I_2, is defined as having one fifth of the image area and the remaining area is divided into four zones with two diagonal lines. If n_j is the number of pixels

in zone j, and R_{ij}, G_{ij}, and B_{ij} are the RGB color components of $\imath$th pixel in zone j,

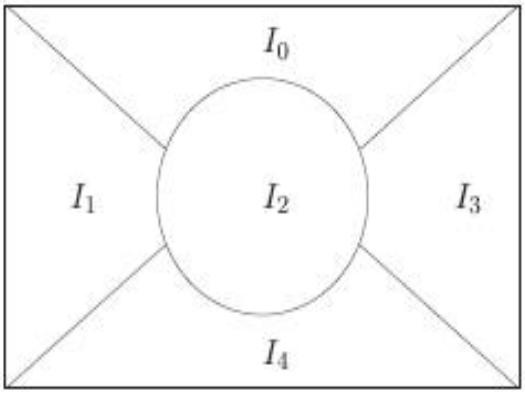

Figure 3.1: The five image zones I_j , j = 0,1,...,4 used when calculating the color j and texture feature vectors.

then the average values are

$$\bar{r_j} = \frac{1}{n_j}\sum_{i=0}^{n_j-1} R_{ij}, \bar{g_j} = \frac{1}{n_j}\sum_{i=0}^{n_j-1} G_{ij}, \bar{b_j} = \frac{1}{n_j}\sum_{i=0}^{n_j-1} B_{ij}, \tag{3.3}$$

The zoning of the image area increases the discriminating power by providing a simple color layout scheme. The result is a 15-dimensional feature vector

$$f_{RGB} = (\bar{r_0}\bar{g_0}\bar{b_0}\bar{r_1}\bar{g_1}\bar{b_1}\bar{r_2}\bar{g_2}\bar{b_2}\bar{r_3}\bar{g_3}\bar{b_3}\bar{r_4}\bar{g_4}\bar{b_4})^T, \tag{3.4}$$

which not only describes the average color of the image but also gives information on the color composition. As an example, Figure 3.2 shows three images and the corresponding image zones I_j colored with the respective average RGB values.

Color: Average YIQ

The YIQ color space is an alternative to the basic RGB color space. It resembles more accurately the way the human brain processes color information. After the transform the Color: Average YIQ feature is calculated similarly in the same five zones I_j =

0,1,...,4 as the Color: Average RGB feature. The YIQ color feature is then

$$f_{YIQ} = (\bar{y_0}\bar{i_0}\bar{q_0}\bar{y_1}\bar{i_1}\bar{q_1}\bar{y_2}\bar{i_2}\bar{q_2}\bar{y_3}\bar{i_3}\bar{b_3}\bar{y_4}\bar{i_4}\bar{q_4})^T, \tag{3.5}$$

in which y_j, i_j, and q_j are the average values of the Y, I, and Q components, calculated in an analogous manner as in 3.5. The dimensionality of Color: Average YIQ feature is thus also 15.

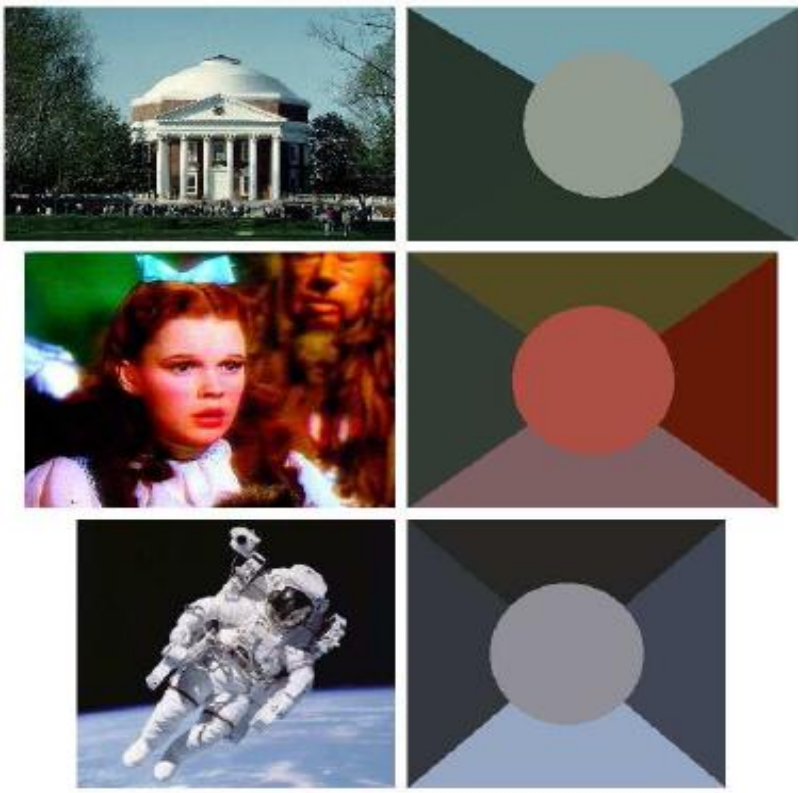

Figure 3.2: Three exemplary images from the ftp.funet.se database and the corresponding image zones colored with average color values.

Texture: Neighborhood

The simple textural feature vectors currently used in SOM are also calculated separately in the same five image zones I_j as the color features. These zones are seen in Figure 3.1. The Y-values (luminance) of the YIQ color representation of every inner

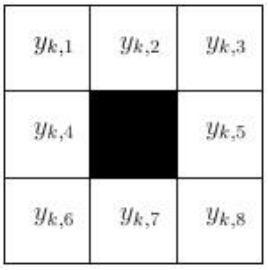

Figure 3.3: The 8-neighborhood of pixel k (shown in black) used to calculate the Texture:Neighborhood feature.

pixel's 8-neighborhood Figure 3.3 are examined and the estimated probabilities $\bar{P_{ij}}$ for each neighbor pixel being brighter than the given center pixel are used as features. If n'_j is the number of border pixels in zone j , y_k is the Y-value of pixel k and $y_{k,i}$ is the Y-value of the neighboring pixel i as shown in Figure 3.3, then the probability estimates $\bar{P_{ij}}$ are calculated as follows:

$$\bar{P_{ij}} = \frac{1}{n_j - n'_j} \sum_{k=0}^{n_j - n'_j - 1} (y_{k,i} > y_k), where(a,b) = \left\{ 1 if a > b, 0 otherwise \right\} \tag{3.6}$$

This results in five eight-dimensional vectors which are combined to one 40-dimensional texture feature vector

$$f_{NBH} = (\bar{p_{0,0}} \bar{p_{0,1}} ... \bar{p_{0,7}} \bar{p_{1,0}} ... \bar{p_{1,7}} \bar{p_{2,0}} ... \bar{p_{4,7}})^T, \tag{3.7}$$

Color regions

A natural solution to add spatial information into the color feature is to divide the images into fixed sub images or zones and compute the features in them separately. This division increases the discriminating power of the feature by providing a color layout scheme. Stricker and Dimai [5] divided images into five partially overlapping,. A more sophisticated approach is to segment the image based on the color variations.

In image segmentation, the image is partitioned into a set of non-overlapping regions. Similarly, in color region extraction or color layout approach, the image is segmented into regions of homogeneous colors. The color region extraction can be performed either automatically or manually. Fully automatic methods are hard to implement as they should reliably separate different objects and still allow small color variations inside the regions. Nevertheless, they are very preferable as manual region extraction can be very time-consuming or even impossible in large databases. The segmented regions may then be represented, e.g. with graphs.

The distances between color pairs in the used color space must correspond to the human perception as visually similar regions should constitute a single region whereas dissimilar regions should be separated. Therefore, the basic RGB color space is considered inadequate for the region extraction approach.

Color histogram

The standard representation for color information in the content-based image retrieval approach has been the color histogram, first investigated in this context by Swain and Ballard (1991)[]. It describes the distribution of colors in an image and is a simple and computationally efficient feature.Most current CBIR systems include some variation of the color histogram.

The one-dimensional histogram **h** of a gray-scale image I with G possible intensity values is a statistical function that provides an estimate of the probabilities of different gray-level values in the image:

$$h_k = \frac{n_k}{n}, k = 0, 1, ..., G - 1, \tag{3.8}$$

where h_k is the kth component of **h**, n is the total number of pixels in the image and n_k is the number of pixels with intensity value k .

The color histogram equivalently denotes the joint probability of the three color channels in the used color space. The color channels in the image are discretized

to a certain number of m_i of intervals. The color histogram of an image is then a $(m_1 * m_2 * m_3)$-sized matrix in which each element contains the number of pixels falling into the corresponding histogram bin. The number of different combinations or histogram bins is thus $M = m_1 m_2 m_3$. Without loss of generality, the color histogram can be regarded as a M -sized vector. In order to simplify the following equations, this notation has been used in the following discussion. The bins of the color histogram can then be considered as color features.

A number of improvements for the standard color histogram have been suggested. One problem is that most color histograms are sparse and sensitive to noise. To increase the robustness of the method, Stricker and Orengo [5] introduced the cumulative color histogram and L_1, L_2, and L_∞ distance metrics to compare them. Other subsequent methods based on the color histogram include color moments, color sets, and the color correlogram. These methods are described later in this section.

An evaluation of different aspects of image retrieval based on the color histogram was made by Zhang [6]. They experimented with different color spaces and color depths, sub image histograms, using only dominant colors, and indexing the histograms with numerical keys. Additionally, they used the Self-Organizing Map to index the feature vectors from the color histograms. A further comparison on color-based techniques, including the cumulative color histogram and color moments.

Different image representations also affect the retrieval performance of the color histogram. To achieve better performance, the gamma values, chromatics of color primaries, and white references should be consistent throughout the whole database and query images. Unfortunately, the currently common image file formats, such as GIF and JPEG, do not capture this information.

Histogram distances

Different metrics for the distance between two color histograms have been proposed. Swain and Ballard (1991) used the Histogram Intersection matching method, which

is essentially the L_1 distance metric of the histograms of two images I and J,

$$d_{L_1}(I, J) = \sum_{k=1}^{M} |(h_k^I - h_k^J)|. \tag{3.9}$$

Another common metric is the L_2 distance function:

$$d_{L_2}(I, J) = \sqrt{\sum_{k=1}^{M} (h_k^I - h_k^J)^2}. \tag{3.10}$$

Yet another distance metric L_{log} , which is non-symmetric and maximizes the log likelihood function, is

$$d_{L_{log}} = \sum_{k=1}^{M} h_k^I \ln h_k^J. \tag{3.11}$$

These metrics are simple and computationally efficient but they may perform poorly in content-based retrieval as some perceptually similar images may have large distances and are thus not retrieved. This happens as the colors may be slightly shifted due to lighting or other conditions and these similarities evade the above metrics, which only compare corresponding histogram bins in the two histograms.
As a part of the QBIC project, Hafner [7] developed a quadratic-form distance metric for color histograms. A general form of the quadratic histogram distance metric is

$$d_{quad} = (h^I - h^J)^T A (h^I - h^J), \tag{3.12}$$

where $\mathbf{A} = [a_{ij}]$ is an (M $*$ M)-sized matrix and a_{ij} denotes the similarity between elements i and j . Quadratic-form metrics thus compare all histogram bins by weighting the inter-bin distances. A naive implementation of the quadratic-form metric is computationally quite expensive, having the complexity O(M^2). With certain pre computations, the complexity can be reduced to O(M).The needed computations are

further reduced by using simple low-dimensional distance measures which are lower bounds of histogram distances [7].

An example of a quadratic-form distance metric is the Mahalanobis distance in which $\mathbf{A} = \Sigma^{-1}$, the inverse of the covariance matrix obtained from a training set of histograms. The Mahalanobis distance is thus given by

$$d_{Mahal}(I, J) = (h^I - h^J)^T \Sigma^{-1} (h^I - h^J). \tag{3.13}$$

Furthermore, other histogram distance metrics have also been developed, such as the Earth Mover's Distance [8]. A comprehensive treatment on measuring histogram similarity along with eight distance metrics can be found in (Smith 1997).

Color moments

In addition to the cumulative color histogram, Stricker and Orengo [5] presented a new method based on color moments in which only the major features of the color distribution are stored instead of the complete distribution. They treated the color distribution of an image like any probability distribution and characterized the distribution by its moments. Then, as most of the information is concentrated in the low-order moments, only the first moment and the second and third central moments of each color channel are used in the color index. The stored values correspond to the average C_i, standard deviation σ_i and the third root of the skewness is s_i of the color channel i, respectively. Suppose the value of the pixel X = (x,y) in the color channel i is $c_i(x, y)$. Then, the index entries of i can be expressed as

$$C_i = \frac{1}{WH} \sum_{x=1}^{W} \sum_{y=1}^{H} c_i(x, y), \tag{3.14}$$

$$\sigma_i = [\frac{1}{WH} \sum_{x=1}^{W} \sum_{y=1}^{H} (c_i(x, y) - C_i)^2]^{\frac{1}{2}} \tag{3.15}$$

$$s_i = [\frac{1}{WH} \sum_{x=1}^{W} \sum_{y=1}^{H} (c_i(x, y) - C_i)^3]^{\frac{1}{3}} \tag{3.16}$$

For image dissimilarity, a weighted Euclidean distance function was used. In their experiments, Stricker and Orengo [5] found the color moments to be more robust and efficient than both the standard and the cumulative color histogram.

Color sets

Smith and Chang [9] proposed color sets as an approximation to the color histogram to facilitate color-based retrieval from large-scale image databases. In the color set approach, a perceptually uniform color space is used and then quantized to M bins.Then m = 0,1,...,M-1 is the index of color (x,y,z) in the used color space assigned by the quantizer function Q_M and given by

$$m = Q_M(x, y, z). \tag{3.17}$$

Now let B^M be an M-dimensional binary space corresponding to the index produced by Q_M so that each axis in B^M corresponds to one color, indexed by m. Usually, the colors m are thresholded so that those colors that are not found frequently enough to a given threshold are discarded. A color set is then a binary vector in B^M describing a set of most prominent colors $\{m_i\}$ in an image or region. Due to the binary property I of color sets, a binary search tree can be constructed to allow fast searching. Color sets are further discussed in (Smith and Chang 1996b)[9]

Color correlogram

One recently suggested replacement for the color histogram is the color correlogram, which incorporates also spatial information into the color features. The color correlogram expresses how the spatial correlation of color pairs changes with distance.In their experiments, Huang et al. found the correlogram to be more robust to small changes in image viewing positions and outperform the traditional color histogram in retrieval accuracy.

3.3 Features Used in Existing Retrieval Systems

The feature types used in the systems are summarized in Figure 3.4. In QBIC, a variety of features can be employed. Depending on the application, the system may support queries based on the average color, color histogram, color layout, based on region descriptors and moment invariants. Virage expresses visual features as image primitives which include the average color, color composition or layout, texture, and object shapes.

The Photo book research project is primarily focused on texture and, therefore, various texture models have been utilized. These include the MRSAR models, Tamura representation, tree-structured wavelet decomposition (TSW) [9], and Wold decomposition. In MARS, the used visual features include the global color histogram, color layout, VisualSEEk enables querying by the color content, sizes and spatial layout of color regions.

The color regions are represented with color sets. In NETRA, the image is segmented into regions and visual features are then computed in the regions.

The current WWW search engines ordinarily utilize less visual features and emphasize the usage of textual information from sources like image file names, URL addresses and HTML tags. In addition, the AltaVista Photo and Media Finder can be used to search for visually similar images based on the visual features used in Virage. In

	QBIC	Virage	Photobook	MARS	VisualSEEk	NETRA	AltaVista	WebSEEk	ImageRover	WebSeer
color	×	×		×	×	×	×	×	×	
color layout	×	×		×	×		×	×	×	
texture	×	×	×	×		×	×		×	
shape	×	×	×	×		×	×			
structure						×				
keywords	×		×	×			×	×	×	×

Figure 3.4: A summary of the feature types used in existent image retrieval systems.

WebSEEk, the textually categorized images can be searched by color histogram and spatial location of color regions. ImageRover supports visual statistics based on color and texture direction histograms. In WebSeer, the images indexed from the World Wide Web were retrieved using textual keywords and face recognition and photograph discrimination algorithms instead of the more generic visual features discussed in this chapter.

3.4 Indexing techniques

Indexing multimedia databases is a different and in many ways more complex problem than indexing traditional databases. The main difficulties arise from the high dimensionality K of the typically used feature vectors. High-dimensional spaces lack many intuitive geometric properties we are accustomed to in low-dimensional spaces. We cannot properly imagine high-dimensional spaces so we try to find low-dimensional analogies where the same effects may not occur. These difficulties are commonly subsumed into the term curse of dimensionality .

In addition, the size of the image database can be large and it may be required to rely on using many features simultaneously in image retrieval. Due to these factors, using basic linear search, where every stored feature vector is considered, easily leads to poor performance. Fast response time is, however, essential in interactive systems as users are quick to reject systems they consider overly sluggish. Therefore, specialized techniques and efficient data structures are needed to manage the retrieval process so that the best-matching images can be determined quickly enough.

A typical task in image retrieval is to determine k nearest data items to a specific point in a high-dimensional feature space, denoted as k-nearest-neighbor (kNN) query. Other types of queries (i.e. point, range, and within-distance queries) are not as important in image retrieval, so the focus in this discussion is on index structures supporting kNN queries. The concept of a nearest neighbor requires a similarity or distance measure. In general, there are two broad categories of index structures for

high-dimensional spaces. The first approach is to apply a divide-and-conquer strategy. The data or the feature space is divided into categories (clusters) or subspaces with the intention that only one or a few of these have to be processed in one given query. Alternatively, we can transform the original feature space into a new space where the operations needed to process a database item are less demanding. This usually means reducing the dimensionality of the original feature space.

3.4.1 Dimensionality reduction

The distribution of image feature vectors in high-dimensional spaces is typically not uniform, but rather has local structure. Also, the features represented at the feature space spanned by the dimensions are often highly correlated, i.e. the intrinsic dimensionality of the data is lower than K. These properties make it feasible to approximate the original space by projecting it into a new space with a lower dimensionality and thus reduced computational requirements. Still, this inevitably results in a loss of information. For kNN queries, the loss of local proximity information between data items is most harmful and should be minimized.

The mapping from a higher-dimensional to a lower-dimensional space, i.e. dimensionality reduction, can be accomplished with linear methods like variable subset selection, principal component analysis (PCA), singular value decomposition (SVD), random projection or nonlinear methods such as multidimensional scaling (MDS) or Self-Organizing Map (SOM).

3.4.2 Clustering

Clustering means partitioning data into m sets or clusters so that data items in a certain cluster are more similar to each other than to data items in other clusters. In the basic form (also called hard or crisp clustering), every data item belongs to exactly one cluster. Clustering can be used to produce an effective image index as follows. After clustering, each cluster is represented by its centroid or sometimes a single rep-

resentative data item (i.e. the image label for that cluster) and, instead of the original data items, the query point is compared to the centroids or the cluster representatives. The best cluster or clusters, according to the used similarity measure, are then selected and the data items belonging to those clusters are evaluated and k nearest neighbors are returned. If the number of clusters is high, we can further cluster the centroids to obtain clusters of clusters, i.e. superclusters, or use some hierarchical clustering method in which the data is gradually clustered from the original data to a single cluster. Many clustering methods have been proposed for image indexing, including competitive learning , the ClusterTree algorithm , agglomerative hierarchical clustering , vector quantization (k-means clustering) , k-medians clustering , and SOM (for a general study on using SOM for clustering).

Chapter 4

Artificial Neural Networks in Classification Problems

An artificial neural network is a system based on the operation of biological neural networks, in other words, is an emulation of biological neural system. Why would be necessary the implementation of artificial neural networks? Although computing these days is truly advanced, there are certain tasks that a program made for a common microprocessor is unable to perform; even so a software implementation of a neural network can be made with their advantages and disadvantages.

Advantages:

- A neural network can perform tasks that a linear program can not.
- When an element of the neural network fails, it can continue without any problem by their parallel nature.
- A neural network learns and does not need to be reprogrammed.
- It can be implemented in any application.
- It can be implemented without any problem.

Disadvantages:

- The neural network needs training to operate.
- The architecture of a neural network is different from the architecture of microprocessors therefore needs to be emulated.
- Requires high processing time for large neural networks..

Another aspect of the artificial neural networks is that there are different architectures, which consequently requires different types of algorithms, but despite to be an apparently complex system, a neural network is relatively simple.
Artificial neural networks (ANN) are among the newest signal-processing technologies in the engineer's toolbox. The field is highly interdisciplinary, but our approach will restrict the view to the engineering perspective. In engineering,neural networks serve two important functions: as pattern classifiers and as nonlinear adaptive filters. We will provide a brief overview of the theory, learning rules, and applications of the most important neural network models. Definitions and Style of Computation

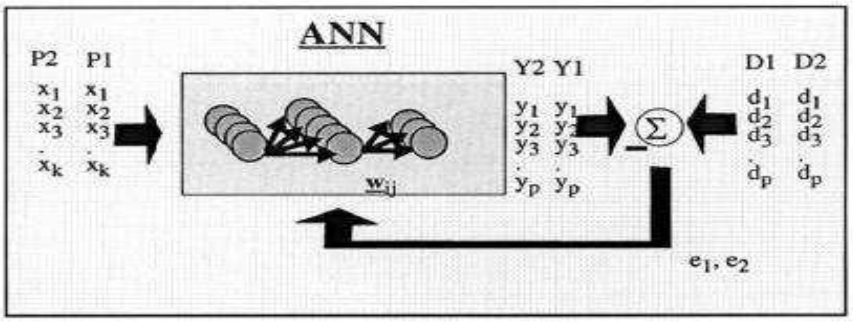

Figure 4.1: ANN.

an Artificial Neural Network is an adaptive, most often nonlinear system that learns to perform a function (an input-output map) from data. Adaptive means that the

system parameters are changed during operation, normally called the training phase. After the training phase the Artificial Neural Network parameters are fixed and the system is deployed to solve the problem at hand (the testing phase).

The Artificial Neural Network is built with a systematic step-by-step procedure to optimize a performance criterion or to follow some implicit internal constraint, which is commonly referred to as the learning rule. The input-output training data are fundamental in neural network technology, because they convey the necessary information to "discover" the optimal operating point. The nonlinear nature of the neural network processing elements (PEs) provides the system with lots of flexibility to achieve practically any desired input-output map, i.e., some Artificial Neural Networks are universal mappers . There is a style in neural computation that is worth describing. An input is presented to the neural network and a corresponding desired or target response set at the output (when this is the case the training is called supervised). An error is composed from the difference between the desired response and the system output. This error information is fed back to the system and adjusts the system parameters in a systematic fashion (the learning rule). The process is repeated until the performance is acceptable. It is clear from this description that the performance hinges heavily on the data. If one does not have data that cover a significant portion of the operating conditions or if they are noisy, then neural network technology is probably not the right solution.

On the other hand, if there is plenty of data and the problem is poorly understood to derive an approximate model, then neural network technology is a good choice. This operating procedure should be contrasted with the traditional engineering design, made of exhaustive subsystem specifications and intercommunication protocols. In artificial neural networks, the designer chooses the network topology, the performance function, the learning rule, and the criterion to stop the training phase, but the system automatically adjusts the parameters. So, it is difficult to bring a priori information into the design, and when the system does not work properly it is also hard to incrementally refine the solution. But ANN-based solutions are extremely

efficient in terms of development time and resources, and in many difficult problems artificial neural networks provide performance that is difficult to match with other technologies.

4.1 Kohenen Neural Network (SOM)

The Kohonen network (Kohonen, 1982, 1984) can be seen as an extension to the competitive learning network, although this is chronologically incorrect. Also, the Kohonen network has a different set of applications. In the Kohonen network, the output units in S are ordered in some fashion, often in a two-dimensional grid or array, although this is application-dependent. The ordering, which is chosen by the user1, determines which output neurons are neighbors. Classification of image segments into a given number of classes using segments features is done by using a Kohonen competitive neural network 4.2. Kohonen networks are feed-forward networks that use an unsupervised training algorithm, and through a process called self-organization, configure the output units into a spatial map. The network contains two layers of

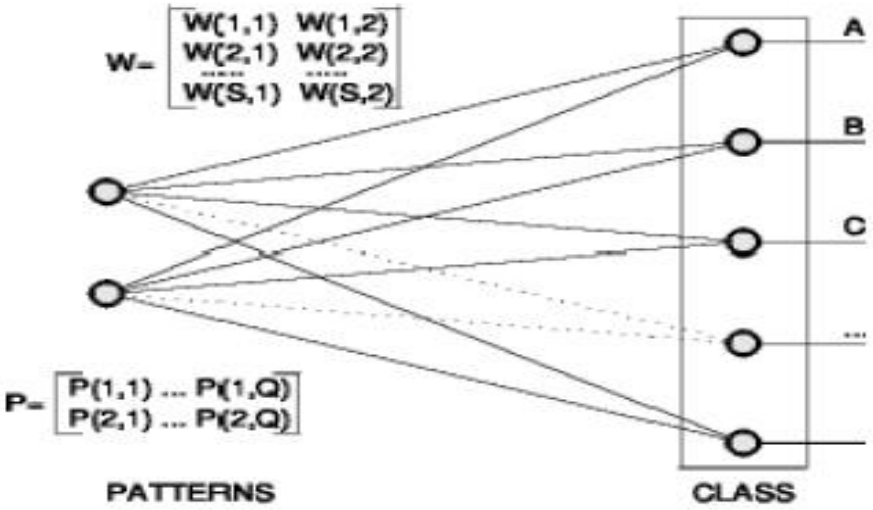

Figure 4.2: Competitive neural network

nodes - an input layer and a mapping (output) layer. Each image is represented by two features in separate columns of the pattern matrix P. The weight matrix W is the connection matrix for the input layer to the output layer. The number of nodes in the input layer is equal to the number of features or attributes associated with the input. The input layer is fully connected to the competitive output layer. The weights are initialized to some chosen small values. Each actual input is compared with each node on the mapping grid.

The winning mapping node is defined as that with the smallest Euclidean distance between the mapping node vector and the input vector. The input thus maps to a given mapping node. The value of the mapping node vector is then adjusted to reduce the Euclidean distance. In addition, all of the neighboring nodes of the winning node are adjusted proportionally. In this way, the two-point input nodes are mapped to a two-dimensional output grid. After all of the inputs of the pattern matrix P are processed (usually after hundreds of repeated iterations), the result should be a spatial organization of the input data organized into clusters of similar (neighboring) regions.

4.2 The SOM Learning Algorithm

During the training period, each unit with a positive activity within the neighborhood of the winning unit participates in the learning process. We can describe the learning process by the equations:

$$wi = a(t)(x - wi)U(Yi) \tag{4.1}$$

Where w, is the weight vector of the ith unit and x is the input vector. The function U(Yi) is zero unless Yi > 0 in which case U(Yi) $= 1$, ensuring that only those units with positive activity participate in the learning process. The factor (t) is written as a function of time to anticipate our desire to change it as learning progresses. To demonstrate the formation of an ordered feature map, we shall use an example in

which units are trained to recognize their relative positions in twodimensional space. Each processing element is identified by its coordinates, (u,v), in twodimensional space. Weight vectors for this example are also two dimensional and are initially assigned to the processing elements randomly. As with other competitive structures, a winning processing element is determined for each input vector based on the similarity between the input vector and the weight vector.

For an input vector x, the winning unit can be determined by $\|x-wc\| = min\|x - wi\|$

Where the index c refers to the winning unit. To keep subscripts to a minimum, we identify each unit in the two-dimensional array by a single subscript. Instead of updating the weights of the winning unit only, we define a physical neighborhood around the unit, and all units within this neighborhood participate in the weight-update process. As learning proceeds, the size of the neighborhood is diminished until it encompasses only a single unit. If c is the winning unit, and N, is the list of

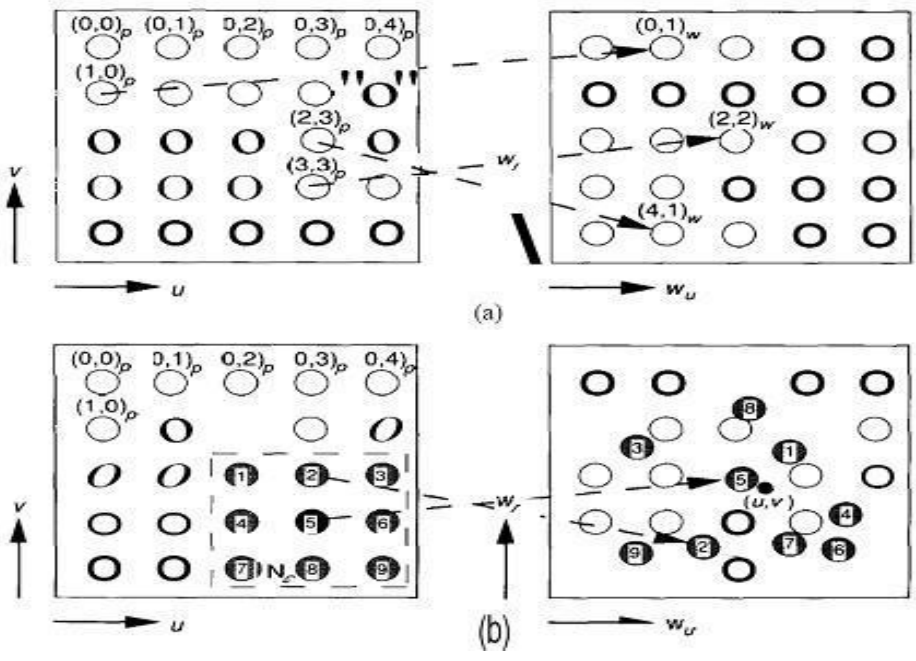

Figure 4.3: weight vector

unit indices that make up the neighborhood, then the weight-update equations are

$$w_i(t+1) = \{^{w_i(t)=\alpha(t)(x-w_i(t))}_{0 \quad otherwise} \quad ^{i\epsilon\ N}, \tag{4.2}$$

Each weight vector participating in the update process rotates slightly toward the input vector, x. Once training has progressed sufficiently, the weight vector on each unit will converge to a value that is representative of the coordinates of the points near the physical location of the unit.

4.3 The Self-Organizing Map

The Self-Organizing Map (SOM) (Kohonen 1997) is an unsupervised, self-organizing neural algorithm widely used to visualize and interpret large high-dimensional data sets. The SOM defines an elastic net of points that are fitted to the distribution of the training data in the input space. It can thus be used to visualize multidimensional data, usually on a two-dimensional grid. Typical applications include visualization of process states or financial results by representing the central dependencies within the data on the map (Kohonen et al. 1996)[1].

The SOM consists of a two-dimensional lattice of neurons. A model vector $m_i \epsilon \Re^n$ is associated with each map unit i.The map attempts to represent all the available observations $x \epsilon \Re^n$ with optimal accuracy by using the map units as a restricted set of models. During the training phase, the models become ordered on the grid so that similar models are close to and dissimilar models far from each other.

The fitting of the model vectors is usually carried out by a sequential regression process, where t = 0,1,2,...,i_{max} - 1 is the step step index: For each input sample x(t), first the index c(x) of the best-matching unit (BMU) or the "winner" model $m_{c_{(x)}}(t)$ is identified by the condition

$$\forall i : \|x(t) - m_{c(x)}(t)\| \leq \|x(t) - m_i(t)\|. \tag{4.3}$$

The usual distance metric used here is the Euclidean one. After finding the BMU, a subset of the model vectors constituting a neighborhood centered around no de c(x) are up dated as

$$m_i(t+1) = m_i(t) + h(t; c(x), i)(x(t) - m_i(t)). \tag{4.4}$$

Here $h(t; c(x), i)$ is the "neighborhood function" , a decreasing function of the distance between the $\imath$th and c(x)th nodes on the map grid. A typical neighborhood function is the Gaussian

$$h(t; c(x), i) = \alpha(t) \exp(-\frac{\|r_i - r_c(x)\|}{2\sigma^2(t)}), \tag{4.5}$$

where $0 < \alpha(\mathrm{t}) < 1$ is the learning rate and σ(t) corresponds to the width of the neighborhood. Both α(t) and σ(t) decrease monotonically as the regression proceeds. The vectors r_i and $r_c(x)$ are the locations of the nodes $\imath$ and c(x) on the SOM surface. A simpler alternative for the neighborhood function is

$$h(t; c(x), i) = \{^{\alpha(t)}_{0} \quad {}^{if \|r_i - r_c(x)\| r(t)}_{otherwise}, \tag{4.6}$$

where r(t) is a monotonically decreasing radius of the neighborhood from node c(x). This regression is reiterated over the available samples and the value of $h(t; c(x), i)$ is let to decrease in time to guarantee the convergence of the prototype vectors m . The I large values of the neighborhood function $h(t; c(x), i)$ in the beginning of the training initialize the network and the small values on later iterations are needed in fine-tuning. SOM consists of neurons organized on a regular lowdimensional grid, see Figure 4.4. Each neuron is a d-dimensional weight vector (prototype vector, codebook vector) where d is equal to the dimension of the input vectors. The neurons are connected to adjacent neurons by a neighborhood relation, which dictates the topology, or structure, of the map. In the Toolbox, topology is divided to two factors: local lattice structure (hexagonal or rectangular, see Figure 4.4) and global map shape (sheet, cylinder or toroid).

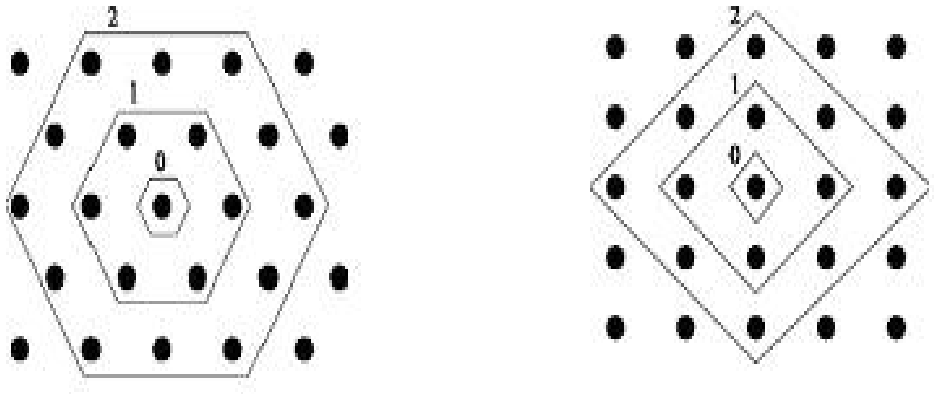

Figure 4.4: Neighborhoods (0, 1 and 2) of the centermost unit: hexagonal lattice on the left, rectangular on the right. The innermost polygon corresponds to 0-, next to the 1- and the outmost to the 2-neighborhood.

The SOM can be thought of as a net which is spread to the data cloud. The SOM training algorithm moves the weight vectors so that they span across the data cloud and so that the map is organized: neighboring neurons on the grid get similar weight vectors.

Two variants of the SOM training algorithm have been implemented in the Toolbox. In the traditional sequential training, samples are presented to the map one at a time, and the algorithm gradually moves the weight vectors towards them, as shown in Figure 4.5 In the batch training, the data set is presented to the SOM as a whole, and the new weight vectors are weighted averages of the data vectors. Both algorithms are iterative, but the batch version is much faster in Matlab since matrix operations can be utilized efficiently. The search for the best-matching neuron dominates the computing time of the SOM algorithm and it can be computationally expensive in high input dimensionality or large SOM networks. The basic algorithm uses full search, where all the neurons must be considered to find the BMU. This makes the complexity of the search O(N), where N is the number of neurons.

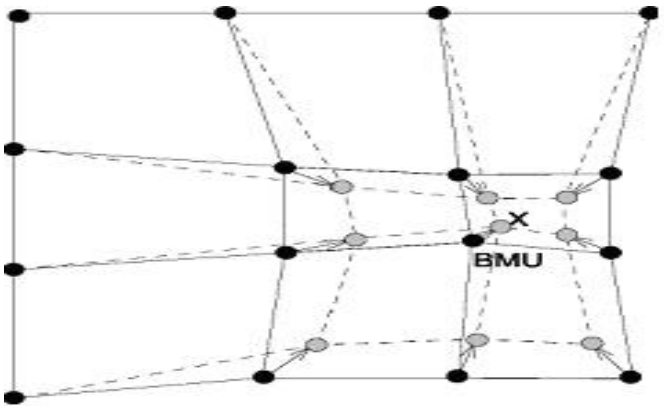

Figure 4.5: Updating the best matching unit (BMU) and its neighbors towards the input sample marked with x. The solid and dashed lines correspond to situation before and after updating, respectively.

4.4 Use of SOM toolbox

4.4.1 Data format

The kind of data that can be processed with the Toolbox is so-called spreadsheet or table data. Each row of the table is one data sample. The columns of the table are the variables of the data set. The variables might be the properties of an object, or a set of measurements measured at a specific time. The important thing is that every sample has the same set of variables. Some of the values may be missing, but the majority should be there. The table representation is a very common data format. If the available data does not conform to these specifications, it can usually be transformed so that it does.

The Toolbox can handle both numeric and categorial data, but only the former is utilized in the SOM algorithm. In the Toolbox, categorial data can be inserted into labels associated with each data sample. They can be considered as post-it notes attached to each sample. The user can check on them later to see what was the meaning of some specific sample, but the training algorithm ignores them. Function *som_autolabel* can be used to handle categorial variables. If the categorial variables

need to be utilized in training the SOM, they can be converted into numerical variables using, e.g., mapping or 1-of-n coding.
Note that for a variable to be "numeric", the numeric representation must be meaningful: values 1, 2 and 4 corresponding to objects A, B and C should really mean that (in terms of this variable) B is between A and C, and that the distance between B and A is smaller than the distance between B and C. Identification numbers, error codes, etc. rarely have such meaning, and they should be handled as categorial data.

4.4.2 Construction of data sets

First, the data has to be brought into Matlab using, for example, standard Matlab functions load and fscanf. In addition, the Toolbox has function *som_read_data* which can be used to read ASCII data files: sD = $som\_read\_data(data.txt)$;
The data is usually put into a so-called data struct, which is a Matlab struct defined in the Toolbox to group information related to a data set. It has fields for numerical data (.data), strings (.labels), as well as for information about data set and the individual variables. The Toolbox utilizes many other structs as well, for example a map struct which holds all information related to a SOM. A numerical matrix can be converted into a data struct with: sD = $som\_data\_struct(D)$. If the data only consists of numerical values, it is not actually necessary to use data structs at all. Most functions accept numerical matrices as well. However, if there are categorial variables, data structs has be used. The categorial variables are converted to strings and put into the .labels field of the data struct as a cell array of strings.

4.4.3 Data preprocessing

Data preprocessing in general can be just about anything: simple transformations or normalization performed on single variables, filters, calculation of new variables from existing ones. In the Toolbox, only the first of these is implemented as part of the package. Specifically, the function *som_normalize* can be used to perform linear and

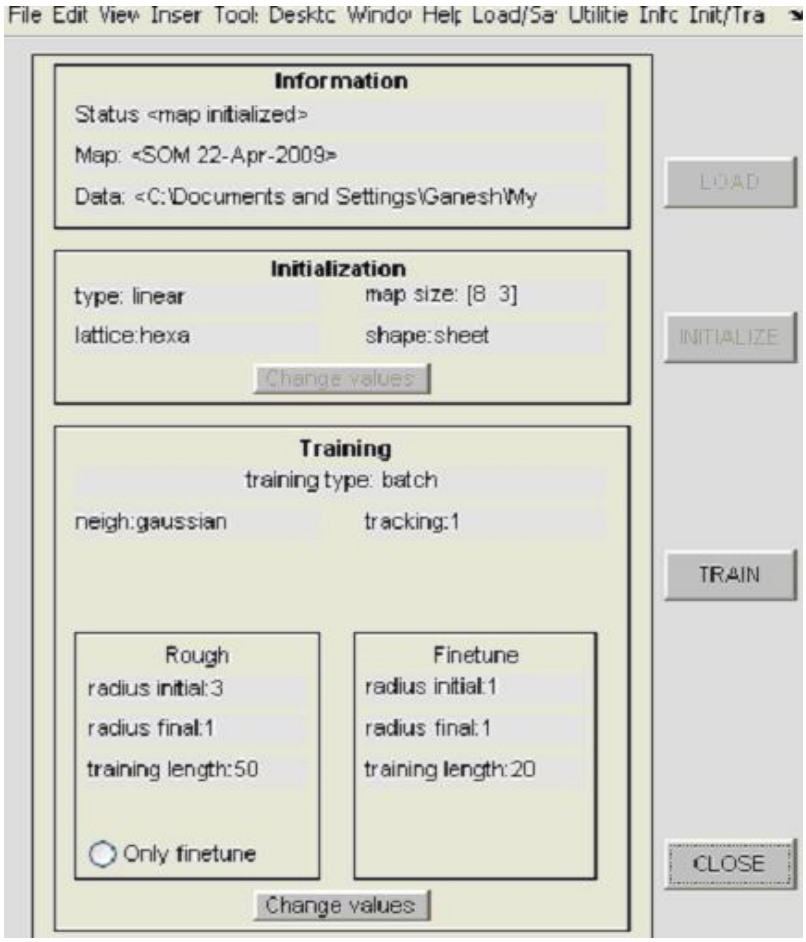

Figure 4.6: Data set preprocessing tool.

logarithmic scalings and histogram equalizations of the numerical variables (the .data field). There is also a graphical user interface tool for preprocessing data, see Figure 4.6.

Scaling of variables is of special importance in the Toolbox, since the SOM algorithm uses Euclidean metric to measure distances between vectors. If one variable has values in the range of [0,...,1000] and another in the range of [0,...,1] the former will almost completely dominate the map organization because of its greater impact on the distances measured. Typically, one would want the variables to be equally important. The standard way to achieve this is to linearly scale all variables so that

their variances are equal to one.

One of the advantages of using data structs instead of simple data matrices is that the structs retain information of the normalization in the field *.comp_norm*. Using function *som_denormalize* one can reverse the normalization to get the values in the original scale: sD = $som\_denormalize(sD)$. Also, one can repeat the exactly same normalization to other data sets.

All normalization are single-variable transformations. One can make one kind of normalization to one variable, and another type of normalization to another variable. Also, multiple normalization one after the other can be made for each variable. The data does not necessarily have to be preprocessed at all before creating a SOM using it. However, in most real tasks preprocessing is important; perhaps even the most important part of the whole process [10].

4.4.4 Initialization and training

There are two initialization (random and linear) and two training (sequential and batch) algorithms implemented in the Toolbox. By default linear initialization and batch training algorithm are used. The simplest way to initialize and train a SOM is to use function *som_make* which does both using automatically selected parameters: $sM = som\_make(sD)$; The training is done is two phases: rough training with large (initial) neighborhood radius and large (initial) learning rate, and finetuning with small radius and learning rate. If tighter control over the training parameters is desired, the respective initialization and training functions, e.g. $som_batchtrain$, can be used directly. There is also a graphical user interface tool for initializing and training SOMs, see Figure 4.7.

4.4.5 Visualization and analysis

There are a variety of methods to visualize the SOM. In the Toolbox, the basic tool is the function *som_show*. It can be used to show the U-matrix and the component

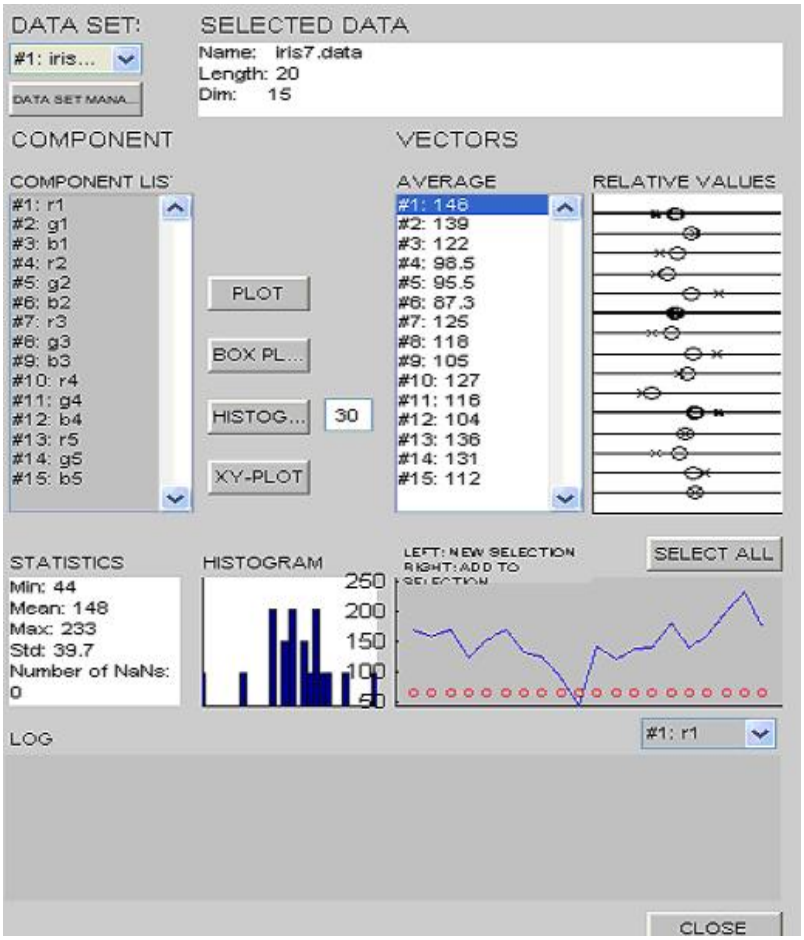

Figure 4.7: SOM initialization and training tool.

planes of the SOM:

$som\_show(sM)$;

The U-matrix visualizes distances between neighboring map units, and thus shows the cluster structure of the map: high values of the U-matrix indicate a cluster border, uniform areas of low values indicate clusters themselves. Each component plane shows the values of one variable in each map unit. On top of these visualizations, additional information can be shown: labels, data histograms and trajectories.

With function *som_vis* much more advanced visualizations are possible. The function is based on the idea that the visualization of a data set simply consists of a set

of objects, each with a unique position, color and shape. In addition, connections between objects, for example neighborhood relations, can be shown using lines. With *som_vis* the user is able to assign arbitrary values to each of these properties. For example, x-, y-, and z-coordinates, object size and color can each stand for one variable, thus enabling the simultaneous visualization of five variables. The different options are:

- the position of an object can be 2- or 3-dimensional.
- the color of an object can be freely selected from the RGB cube, although typically indexed color is used.
- the shape of an object can be any of the Matlab plot markers ('.','+', etc.), a pie chart, a bar chart, a plot or even an arbitrarily shaped polygon, typically a rectangle or hexagon.
- lines between objects can have arbitrary color,width and any of the Matlab line modes, e.g. '-'.
- in addition to the objects, associated labels can be shown.

For quantitative analysis of the SOM there are at the moment only a few tools. The function *som_quality* supplies two quality measures for SOM: average quantization error and topographic error.

Chapter 5

Implementation and Results

5.1 Implementation

5.1.1 Features

SOM may use either one or several types of statistical features simultaneously for image querying. Separate feature vectors can be formed for describing, for example, the color content, texture, shape, and structure of the image. With a wider definition, features can also include textual descriptions and other available information about the images, in addition to the visual features extracted directly from the image content.

A separate SOM is constructed for each visual feature and the results from the maps are combined to find the most similar images to be presented to the user.

5.1.2 Visual Feature Extraction

Currently, the following experimental feature has been implemented.

Color:AverageRGB

The average values for red, green, and blue color channels ($\bar{r_j}$, $\bar{g_j}$, $\bar{b_j}$) are calculated in the five separate zones I_j, j = 0,1,..., 4 of the image as seen in Figure 5.1. The circular zone, I_2, is defined as having one fifth of the image area and the remaining

area is divided into four zones with two diagonal lines. If n_j is the number of pixels in zone j, and R_{ij}, G_{ij}, and B_{ij} are the RGB color components of $\imath$th pixel in zone j,

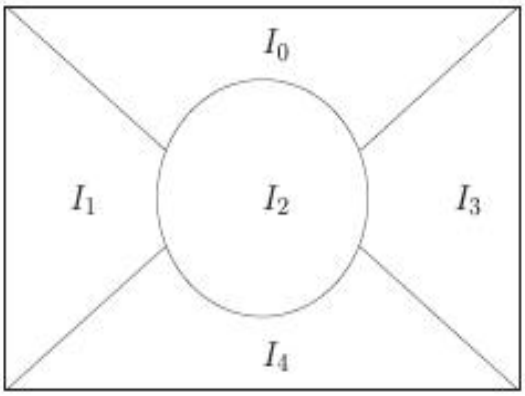

Figure 5.1: The five image zones I_j , j = 0,1,...,4 used when calculating the color j and texture feature vectors.

then the average values are

$$\bar{r_j} = \frac{1}{n_j}\sum_{i=0}^{n_j-1} R_{ij}, \bar{g_j} = \frac{1}{n_j}\sum_{i=0}^{n_j-1} G_{ij}, \bar{b_j} = \frac{1}{n_j}\sum_{i=0}^{n_j-1} B_{ij}, \tag{5.1}$$

The zoning of the image area increases the discriminating power by providing a simple color layout scheme. The result is a 15-dimensional feature vector

$$f_{RGB} = (\bar{r_0}\bar{g_0}\bar{b_0}\bar{r_1}\bar{g_1}\bar{b_1}\bar{r_2}\bar{g_2}\bar{b_2}\bar{r_3}\bar{g_3}\bar{b_3}\bar{r_4}\bar{g_4}\bar{b_4})^T, \tag{5.2}$$

which not only describes the average color of the image but also gives information on the color composition. As an example, Figure 5.2 shows three images and the corresponding image zones I_j colored with the respective average RGB values.

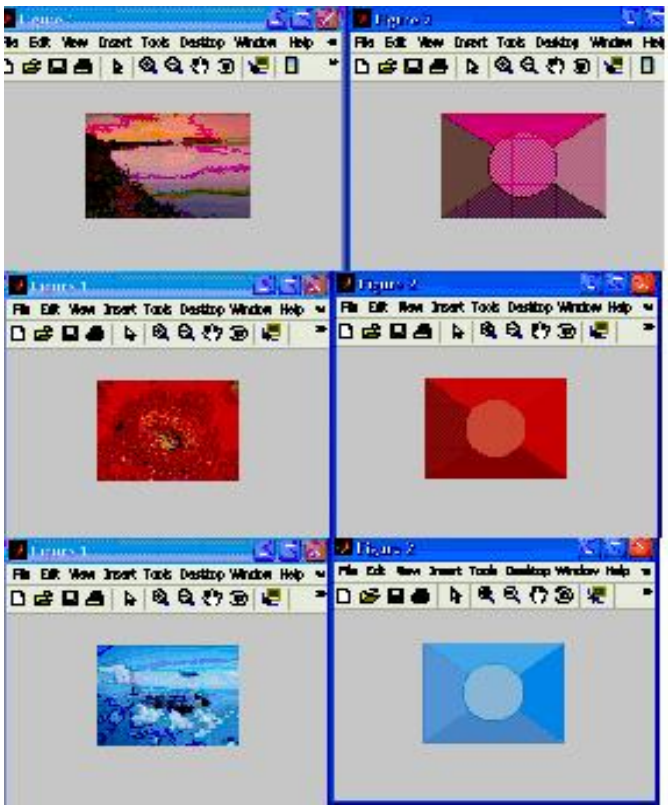

Figure 5.2: Three exemplary images from the database and the corresponding image zones colored with average color.

5.2 Experiments

5.2.1 Data Sets

For experimentation three categories of images are used namely **Rose**, **Sky**, **Sunset** images.To understand the performance of the SOM 5 data sets are formed out of these three categories of images.In the first data set 10 images of each category are used and performance in terms of number of misclassification is studied.Gradually the number of images in each category is increased to 20, 30, 40, and 50 in the later data set and the performance of the SOM toolbox for image classification is studied.

5.2.2 Data set 1

In data set 1, 30 images are used, in which 10 images are belong to sunset class, 10 images are rose images, and another 10 images are sky images.5.3 The data is in an

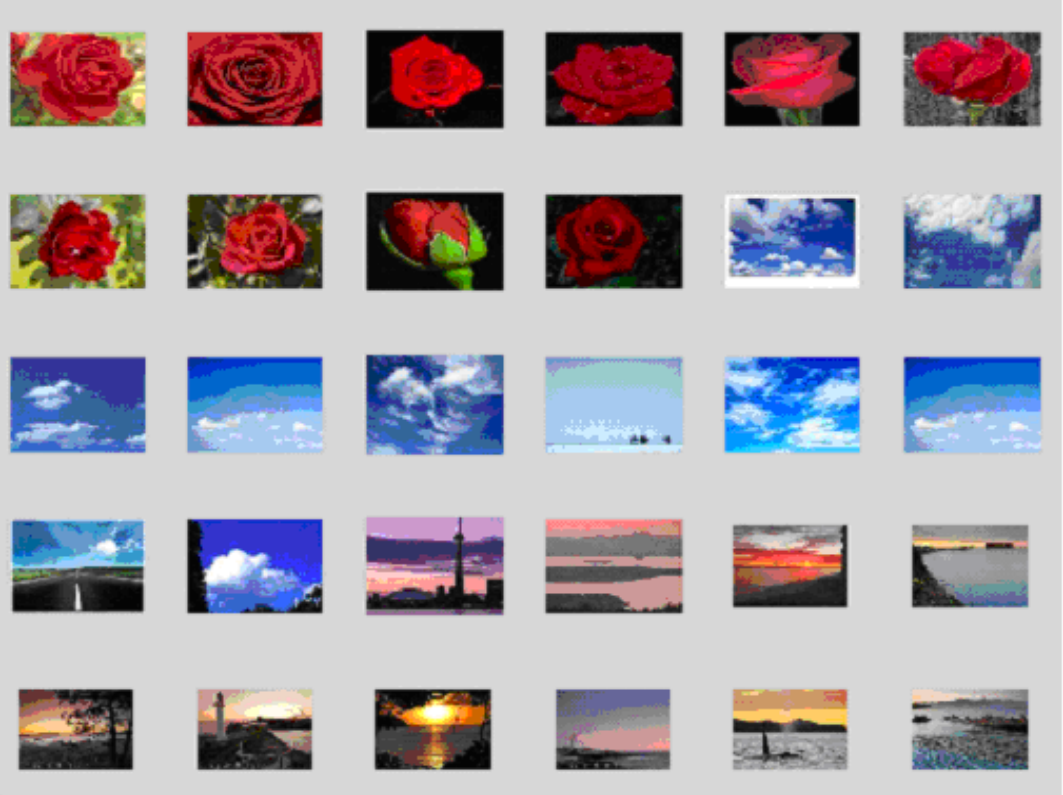

Figure 5.3: Three exemplary images from the database as a Data set 1

ASCII file, the first few lines of which are shown below. The first line contains the names of the variables. Each of the following lines gives one data sample beginning with numerical variables and followed by labels.

♯n r1 g1 b1 r2 g2 b2 r3 g3 b3 r4 g4 b4 r5 g5 b5

191 80 62 142 78 45 139 60 40 180 113 75 165 19 29 rose1

154 167 197 207 215 231 136 155 195 92 121 179 81 112 180 sky1

78 53 22 70 51 31 65 47 29 86 52 24 173 135 74 sunset1

...

First of all the data set is loaded into Matlab by using *som_read_data*. After the data set is ready, a SOM is trained by using *som_make*. Since the data set had labels, the map is also labeled using *som_autolabel*. After this, the SOM is visualized using *som_show*.

Read data from an ascii file

$sD = som\_read\_data('sample1.data');$

Make the SOM

$sM = som\_lininit(sD,'msize',[10 \quad 10]);$

$sM = som\_batchtrain(sM, D,'radius',[31],'trainlen',300,'gaussian');$

Basic visualization

$som\_show(sM, umat, all, comp, 1:4, ...empty, Labels, norm, d);$

$som\_show\_add('label', sM);$

Automate clustering using K-means algorithm

$[c, p, err, ind] = kmeans\_clusters(sM);$

$[dummy, i] = min(ind);$

$som\_show(sM,'color', pi, [int2str(i),'clusters'])$

- (*SOM_MAKE*)- this function create, initialize and train Self-Organizing Map. First, the map size is determined. After that SOM is initialized. There is two algorithms for initialization, first linear initialization along two greatest eigenvectors is tried, but if this can't be done (the eigenvectors cannot be calculated), random initialization is used instead.

  After initialization, the SOM is trained in two phases: first rough training and then fine-tuning. If the 'tracking' argument is greater than zero, the average quantization error and topographic error of the final map are calculated.

  The feature map is shown in Figure 5.4.The U-matrix is shown along with all fifteen component planes.First the Unified distance matrix (U-matrix) calculated based on all feature vector and then the component planes. U-matrix is common way to represent information about the map.Each component plane

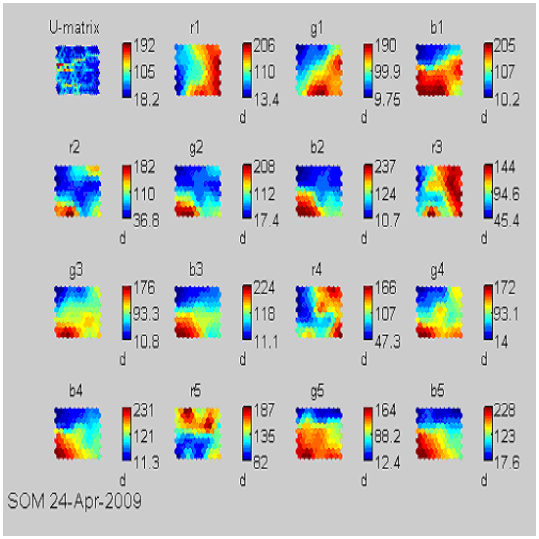

Figure 5.4: Feature map. Umatrix on top left, then component planes.

shows the values of one variable in each map unit.Here hexagonal map structure was used.

Experiments were done with different sizes of maps. Size of the map mostly influence on scaling factor of representing the output space. After series of experiments that default size of the map (10∗10) gives quite good representation of results.

- ($SOM\_AUTOLABEL$)- this function automatically labels given map/data struct based on an already labeled data/map struct. Basically, the BMU of each vector in the sFrom (data or map struct from which the labels are taken) is found from among the vectors in sTo(data or map struct to which the labels are put,the modified struct is returned), and the vectors in sFrom are added to the corresponding vector in the sTo struct. After labeled we got this result as shown in Figure 5.5

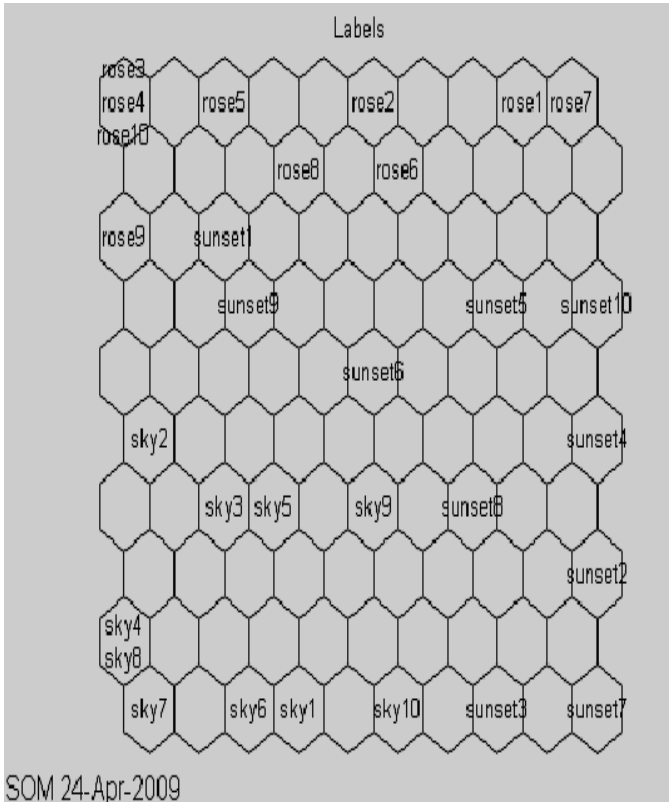

Figure 5.5: $SOM\_AUTOLABEL$.

- ($KMEANS\_CLUSTERS$)- by using this function we can Clustering with k-means with different values for k.Makes a k-means to the given data set with different values of k. The k-means is run multiple times for each k, and the best of these is selected based on sum of squared errors. Finally, the Davies-Bouldin index is calculated for each clustering.Figure 5.6.And after $som\_show\_add('label', M)$ we got result as shown in Figure 5.7

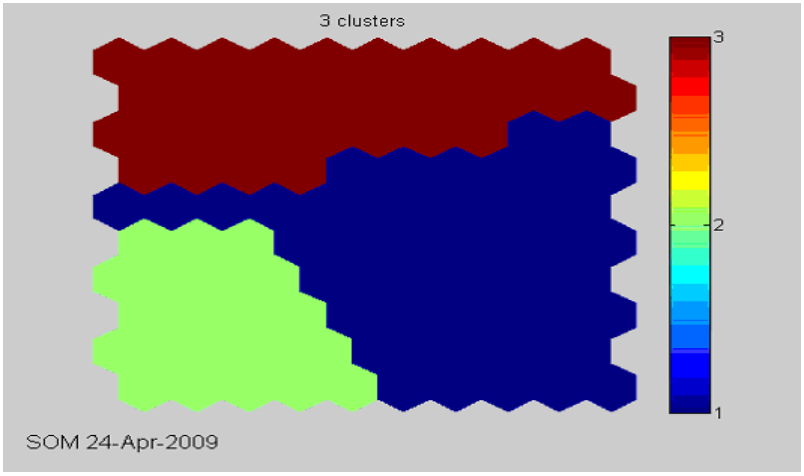

Figure 5.6: Davies-Bouldin based clustering.

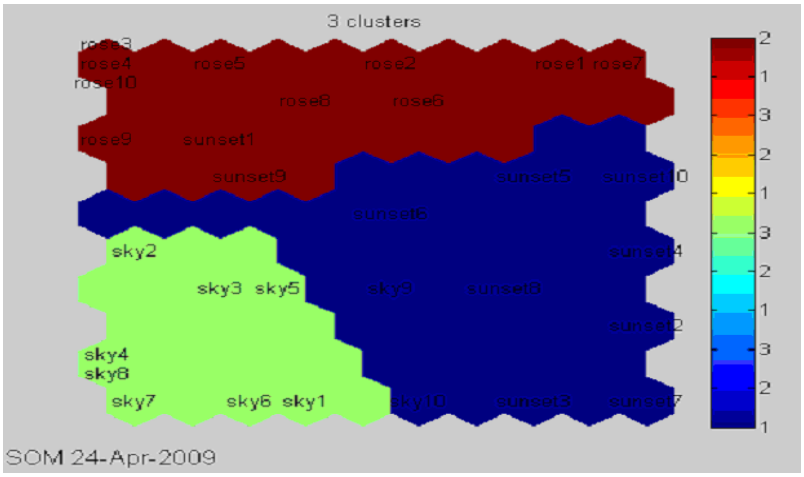

Figure 5.7: After auto label SOM based clustering.

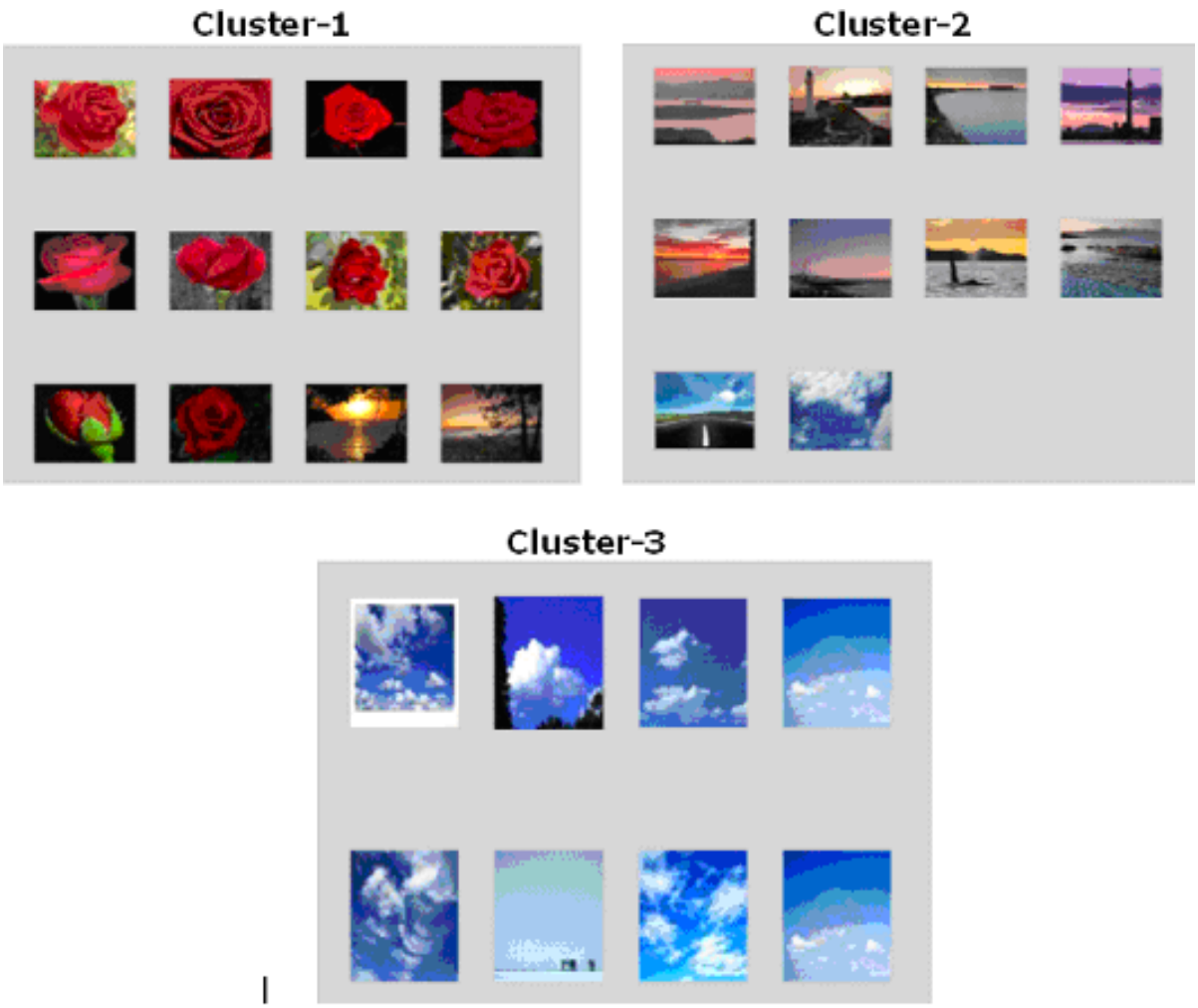

Figure 5.8: After Clustering three exemplary images from the database.

After clustering it is categorized in three groups as shown in Figure 5.8.In this data set out of thirty images, four images are wrongly classified.

5.2.3 Data set 2

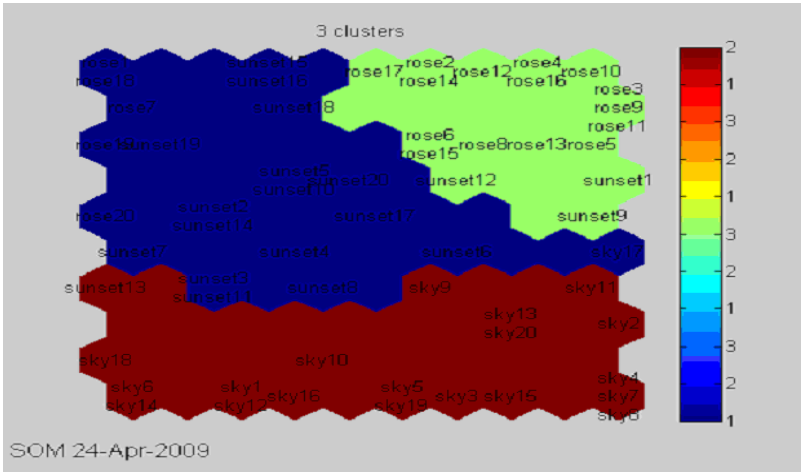

Figure 5.9: SOM clustering of Data set 2.

5.2.4 Data set 3

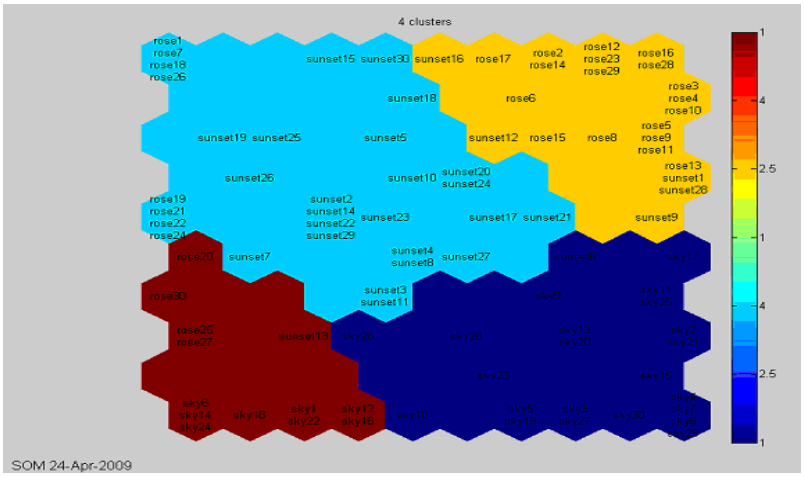

Figure 5.10: SOM clustering of Data set 3.

5.2.5 Data set 4

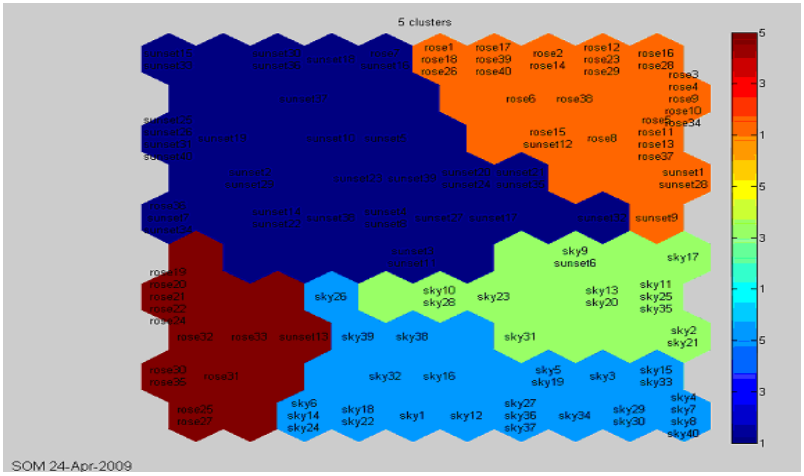

Figure 5.11: SOM clustering of Data set 4.

5.2.6 Data set 5

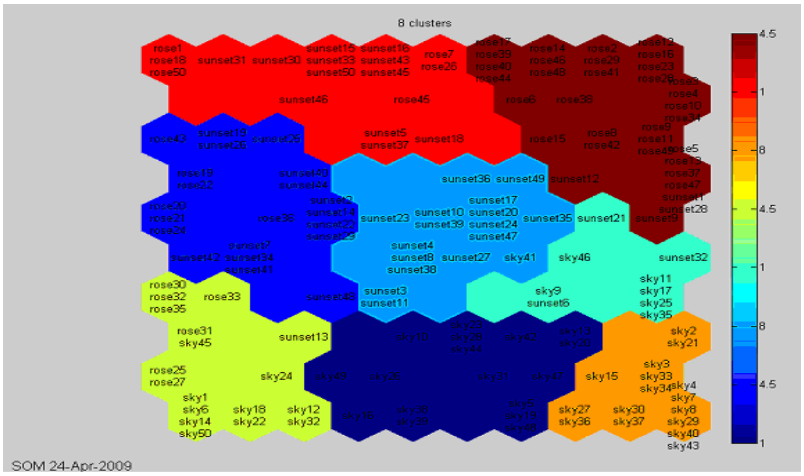

Figure 5.12: SOM clustering of Data set 5.

5.3 Analysis

In **first data set** out of thirty images, four images are wrongly classified.In **second data set** out of sixty images, ten images are wrongly classified.In **third data set** actual there is three classes, but it clusterings it into four classes. Sky images grouped into two classes .Out of ninety images, seventeen images are wrongly classified.

In **fourth data set**, as increases the number of images in each category, number of cluster also increases.Sky images and rose images grouped into two classes instead of one. Out of one hundred twenty images, seven images are wrongly classified.And in **fifth data set** it clusterings into eight classes. Sunset grouped into three classes and sky images grouped into four classes instead of one.Out of one hundred fifty images, twenty images are wrongly classified.

Efficiency are given in 5.3.

No. of Data set	No. of images	No. of actual classes	categorized into No. of classes	Efficiency in %
Data set 1	30	3	3	86.66%
Data set 2	60	3	3	83.33%
Data set 3	90	3	4	81.11%
Data set 4	120	3	5	94.11%
Data set 5	150	3	8	86.66%

Table I: Analysis for different data set

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

The classified images will help in restricting in the search area of content-based image retrieval (CBIR) to only the cluster the query image belongs to or nearby by clusters. In this thesis, the SOM Toolbox has been shortly introduced. The SOM is an excellent tool in the visualization of high dimensional data [2]. As such it is most suitable for data understanding phase of the knowledge discovery process, although it can be used for data preparation, modeling and classification as well.

Image indexing with the SOM was perceived to be a robust and effective solution which tolerates even very high input vector dimensionality. As an indexing method, the SOM was interpreted as a combination of clustering and dimensionality reduction. Unlike basic clustering algorithms, the SOM has the advantage of providing a natural ordering for the clusters due to the preserved topology. This way, the relevance information obtained from the user can be spread to neighboring image clusters.

One drawback is that the SOM algorithm is linked with the Euclidean distance measure in its basic form. Euclidean distance was thus used with all features. For certain features this probably is suboptimal as for example the MPEG-7 Descriptors have their own distance measures defined. Specific treatment for each feature separately

could result in performance increase, although this would require a more complex training algorithm.

CBIR is a valid way to query huge databases that are quite common nowadays. CBIR methods give us efficient tools to manage, and analyze data. Image relevance or semantics cannot be objectively defined and the correct action of a retrieval system is always context and user-dependent. This makes designing and especially evaluating automatic tools for CBIR a challenging task.

6.2 Future Scope

- The feature selection is not restricted, so in the image the feature vector can be expanded to include textural and shape information.As long as an equal number of features are calculated from each image in the database.

- Image class can be expanded to multimedia messages which seems to be promising area. The feature vector in such information would require features for sound along with visual informations.Combining content-based features with textual annotations can help in improving the performance of clustering.

Appendix A

MatLab scripts

clear all; close all; echo on;

$sD = som\_read\_data('data.txt');$

$sM = som\_lininit(sD,'msize',[10 \quad 10]);$

$sM = som\_batchtrain(sM, D,'radius',[31],'trainlen',300,'gaussian');$

$[qe, te] = som\_quality(sM, D);$

$som\_show(sM,'umat','all','comp','all');$

$som\_show(sM,'empty','Labels');$

$sM = som\_autolabel(sM, D);$

$som\_show\_add('label', sM);$

$[c, p, err, ind] = kmeans\_clusters(sM);$

$[dummy, i] = min(ind);$

$som\_show(sM,'color', pi, [int2str(i),'clusters'])$

References

[1] T. Kohonen, *Self-Organizing Maps,*. Springer-Verlag, 2001.

[2] Laboratory of computer and information science, *SOM Toolbox*. http://www.cis.hut.fi/projects/somtoolbox.

[3] L. J. K. M. Oja, E. and S. Brandt, "Self-organizing maps for content-based image retrieval," *Kohonen Maps, Elsevier*, 1999.

[4] S. Brandt, "Use of shape features in content-based image retrieval," tech. rep., Helsinki University of Technology., 1999.

[5] M. Stricker and A. Dimai, "Color indexing with weak spatial constraints, storage and retrieval for image and video databases," *SPIE Proceedings Series, San Diego*, 1996.

[6] H. Zhang and D. Zhong, "A scheme for visual feature based image indexing, storage and retrieval for image and video databases," *SPIE Proceedings Series*, 1995.

[7] S. H. S. E. W. F. M. Hafner, J. and W. Niblack, "Efficient color histogram indexing for quadratic form distance functions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.

[8] T. C. Rubner, Y. and L. J. Guibas, "The earth movers distance as a metric for image retrieval," tech. rep., Stanford University, 1998.

[9] N.-S. Chang and K.-S. Fu, "Query-by-pictorial-example," *IEEE Transactions on Software Engineering*, 1980.

[10] P. D., *Data Preparation for Data Mining.* Morgan Kaufman Publishers,, 1999.