IMAGE SEARCHING WITH SIGNATURE FILTERING AND MULTIDIMENSIONAL INDEXING

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER
ENGINEERING AND INFORMATION SCIENCE
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Çağlar Günyaktı
July, 1997

IMAGE SEARCHING WITH SIGNATURE FILTERING AND MULTIDIMENSIONAL INDEXING

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING AND INFORMATION SCIENCE AND THE INSTITUTE OF ENGINEERING AND SCIENCE OF BILKENT UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

Çağlar Günyaktı

By

Çağlar Günyaktı

July, 1997

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. M.EROL ARKUN(Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. David Davenport

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Dr. Seyit Koçberber

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet Baray, Director of Institute of Engineering and Science

ABSTRACT

IMAGE SEARCHING WITH
SIGNATURE FILTERING
AND
MULTIDIMENSIONAL INDEXING

Çağlar Günyaktı
M.S. in Computer Engineering and Information Science
Supervisor: Prof. Dr. M.Erol ARKUN
July, 1997

The content of multimedia information is conducive to variable and subjective interpretation which makes indexing and content-based searching a difficult task. This thesis addresses such image database issues as performance degradation problem in indexing with the increase in the number of dimensions, query interfaces for efficient and effective querying and content-based feature categorization. In particular, image feature representation, content-based image retrieval and multi-dimensional indexing for efficient searching are surveyed. A different approach for content-based querying is proposed and a prototype of an image search engine, called *SIS* (**S**ignature based **I**mage Filtering and **S**earch), that is accessible via Internet, is implemented using the subset of the proposed solutions. In SIS, image signatures are calculated using basic image features (color, shape and texture). These signatures describe not only the image content as a whole, but also the subobjects and their orientations residing in the image. Signatures are used for filtering the search space, by employing a multidimensional indexing structure known as *TV-tree*. SIS utilizes basic image feature queries and reports back the matching features to help accelerate the navigation towards the required visual information. A content-based search via SIS's user interface is additionally augmented with keyword-based queries to facilitate searching by criteria which are impossible to specify by image features alone.

Key words: Image databases, content-based querying, multi-dimensional indexing, image signatures, Information Retrieval

ÖZET

İMZA FİLTRELEME ve ÇOK BOYUTLU İNDEKSLEMEYLE ŞEKİL ARAMA

Çağlar Günyaktı
Bilgisayar ve Enformatik Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. M.Erol ARKUN
Temmuz, 1997

Çoklu ortam bilgilerinin içeriği değişken ve öznel yorumlamaya yatkındır. Bu durum indeksleme ve içeriğe dayalı aramayı zorlaştırır. Bu tezde, artan boyut sayısıyla düşen performans sorunu, yeterli ve etkili sorgulama için sorgu arayüzleri ve içerik özelliklerinin sınıflandırılması gibi şekil içerikli veri tabanlarının sorunları araştırılmıştır. Özellikle, şekillerinin özelliklerinin ifadelendirilmesi, içeriğe dayalı şekil sorgulanması ve etkili arama için çok boyutlu indeksleme yapıları araştırılmıştır. İçeriğe dayalı sorgulama için farklı bir bakış açısı önerilmiş ve İnternetten ulaşılabilinen *İŞA* (**İ**mzaya dayalı **Ş**ekil filtrelenmesi ve **A**raması/Signature based image filtering and search) adlı prototip şekil arama mekanizması önerilen çözümlerin bir kısmı kullanılarak gerçekleştirilmiştir. İŞA'da şekil imzaları, temel şekil özellikleri (renk, biçim, ve dokum) kullanılarak hesaplanır. Bu imzalar sadece şekil içeriğini bir bütün olarak değil alt nesneler ve onların şekil içindeki konumlarını da tanımlar. İmzalar, *TV-ağacı* diye bilinen çok boyutlu bir indeksleme yapısı kullanılarak arama uzayının filtrelenmesi ve taranması için kullanılır. İŞA şekillerin temel özelliklerinin sorgulanmasını sağlar ve istenen görüntüsel bilgilere erişimi hızlandırmaya yardım etmek için eşlenen özellikleri kullanıcıya geri bildirir. İŞA'nın kullanıcı arayüzü kullanılarak yapılan içeriğe dayalı arama yeteneği, şekil özelliklerinin tek başlarına kesinlikle belirtmelerinin imkansız olduğu kriterlere göre de aranmasını sağlamak için ek olarak anahtar kelime tabanlı sorgulama desteği ile artırılmıştır.

Anahtar kelimeler: Şekil Veri Tabanları, içeriğe dayalı sorgulama, çok boyutlu indeksleme, şekil imzaları, Bilgi Erişimi.

To my parents

ACKNOWLEDGMENTS

I am very grateful to my supervisor, Prof. Dr. M.Erol ARKUN for his invaluable guidance and motivating support during this study. His instruction will be the closest and most important reference in my future research. I would also like to thank Dr. Reda Al-Hajj for his guidance, my girl-friend Çiğdem and my colleague Ozan Özhan who always provided me with moral support, Assist. Prof. Kin-Ip (David) Lin, H.V. Jagadish and Christos Faloutsos for providing me the original code of TV-tree for integrating with my implementation, and again King-Ip Lin for his technical support during the integration, my friends Yücel Saygın, Ferit Fındık for their help whenever I met with obstacles, my family for their moral support and patience during the stressful moments of my work, and last but not the least, Tahsin Mertefe Kurç, who was always ready for help with his priceless technical knowledge and experience.

Finally, I would like to thank the committee members Asst. Prof. Dr. David Davenport and Dr. Seyit Koçberber for their valuable comments, and everybody who has in some way contributed to this study by lending moral and technical support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Database management systems are used for maintenance and manipulation of huge amounts of structured information.The main advantage of storing data in a database is the applicability of database features, such as data independence (data abstraction), openness (application neutrality), high-level querying/retrieval facility, concurrency control, persistency and crash recovery (fault tolerance).

Traditional databases support a number of basic data types, such as character strings, integers, floating point numbers, briefly alphanumeric data types, in relational tables or in object-oriented fashion.

Developing technology and emerging needs change the type and volume of information flow available before. The type of information is not restricted to only alphanumeric data, which is generally called textual data. The new, non-alphanumeric data such as audio and visual data, briefly multimedia data, were introduced more than a decade ago.

Multimedia is probably one of the most over-used terms of the early 1990s. There is no single accepted definition for multimedia data, however it is safe to state that it is the *combination of different* data types. The elements of multimedia data are text, graphics, images, audio, animation and video data (Figure 1.1).

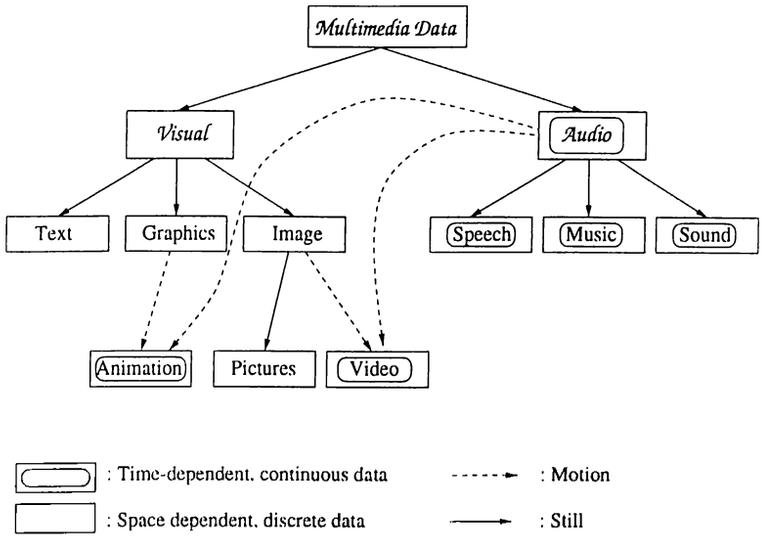

Figure 1.1: Multimedia Data Hierarchy

Multimedia data is separated from traditional (alphanumeric) data in the following respects;

1. *Data types (e.g new image & speech formats)*
2. *Volume of data (e.g. a 600*600*256 color image is 2.8Mbytes)*
3. *Time dependency and synchronization (e.g. audio, video data)*
4. *Storage requirements for continuous data flow (e.g. video data)*
5. *Indexing structures (e.g. multi-dimensional tree structures)*
6. *Query and retrieval patterns (e.g. content-based retrieval and similarity functions)*

Therefore the introduction of multimedia data has opened up new research areas [Kim, Gha95] during the last decade. Specifically, such research is related with applications such as computer networks, distributed computing [GBD+94], data compression, computer graphics, pattern/voice recognition, machine learning, user interfaces, computer hardware, artificial intelligence and databases.

Multimedia databases (MMDBs) are high capacity/performance DBMSs that support multimedia data types as well as basic alphanumeric types. MMDBs

are still in their infancy compared to the work on multimedia applications and multimedia data types. The mentioned database features are hardly ever supported by existing systems.

While conventional DBMSs offer many useful and powerful facilities for searching and indexing, they are not well-suited for non-alphanumeric data [BA95, AK92]. Although some systems try to solve this problem by representing the multimedia data as Binary Large Objects (BLOBs) which is associated with its display software, they still have the following problems [AK92]:

(a) the restrictiveness of available operations (blob_read, blob_write)
(b) absence of a querying facility, because they are just like a data repository.

Furthermore, only a few systems support a limited form of content-based queries.

1.1 Image Databases

An *image database management system* is a database system that stores images and supports the data type image with a set of functions, such as transformation from and to different file formats, change of color depth, content-based retrieval, tiling regions of interest and automatic indexing. However, there exists hardly any commercial image database management system. It is quite common to store images in the file system and to use the database only for links and administrative data, i.e. the images themselves are not really part of the database; they are only referenced by text-strings or pointers.

The image database inherits some of the problems of multimedia databases and some others, such as image information representation, querying mechanism, data modeling and formatting. Except some write accesses, i.e. adding new images, and updating index information, image databases are mostly accessed read-only. Hence an efficient query mechanism becomes an important factor for image databases.

This thesis focuses on image databases and proposes solutions for such existing problems as performance degradation problem in indexing with the increase in the number of dimensions, query interfaces for efficient and effective querying and content-based feature categorization. In particular, image feature representation, content-based image retrieval and multi-dimensional indexing for efficient searching are surveyed. A different approach for content-based querying is proposed and a prototype of an image search engine[1], that is accessible via Internet[2] is implemented using the subset of the proposed solutions. The importance of the marriage of database technology and Information Retrieval (IR) is emphasized to overcome the mentioned problems.

1.1.1 Content–based Queries

Query by image content is the searching of images based on the common, intrinsic and high-level properties such as color, texture, shape of objects captured in the images and their semantics. Content-based searches are performed directly on image data as opposed to searches of associated textual information that has been attached to each database image by an interpreter or analyst.

Content-based retrieval greatly improves the value of massive image databases, where huge amounts of new information should be searched within a short time. Visual information is unlike text due to its condensed and abstract meaning. This is the main reason for the content-based indexing problem [BA95]. Especially, the old cliche *"A picture is worth a thousand words"*, explains why image data cannot be handled with traditional methods. A complete annotation of an image with $\mathbf{n}$ objects each with $\mathbf{m}$ attributes requires $\mathbf{O(n^2m^2)}$ database entries, because each one of the $\mathbf{n}$ objects may have $\mathbf{m}$ number of attributes, so $O(nm)$ database entries are needed, the number of entries are increased to $O(n^2m^2)$, when their spatial relationship are included. If the interrelationships between images are also included, the indexing problem becomes intractable [PPS94].

Content-based queries related with alphanumeric data types are generally

[1] SIS-Signature based Image Filtering and Search.
[2] http://www.nlp.cs.bilkent.edu.tr/~gunyakti/INTERFACE.html

straightforward. The queries contain some predicates which must be satisfied by any data that is retrieved. For example, queries such as "*find all students who failed in CS352*", and range queries such as "*find all students whose GPA is between 3.00 and 3.49*" are *exact match* queries. However, querying multimedia data is different from conventional query scheme. Multimedia data requires its contents to be interpreted for querying. Therefore, content-based queries require sophisticated indexing schemes and content-analysis algorithms to generate content descriptions.

In image databases, image content description or content-based retrieval of images is still in its infancy. It exploits many different areas from computer vision to Natural Language Processing (NLP) [Sri95], from machine learning to IR. The need for a domain expert is rather explicit in content-based interpretation of images. However, some of the basic image features such as color, texture, shape and their spatial orientations are perceived less subjectively, e.g., the sun is interpreted as a circular shaped, yellow or orange colored object using the basic features, whereas it may indicate power, clarity or daylight, according to the scene or scope of the image. Briefly, one can say that the content of visual information is conducive to variable, subjective interpretation.

Therefore these basic image features, which are high-level abstractions of visual information, are used for describing the contents of images. The spatial location of image objects, relationships among them, their shapes, and color distributions are all exploited in content-based queries. For example, "*Retrieve images that contain such a texture, with a blue circular object adjacent to a red, rectangular object on a blue background*". However, text-based queries actually answer the questions "*who, where, how, why ?*" related to the event captured by the image.

Another problem associated with content-based queries is the selection of features. In medical images, sometimes color and their spatial locations are important. For example, a vast amount of color brain tomographies should be scanned according to the mentioned features while conducting a survey on phases of brain tumors in the frontal lobe. Other times the texture pattern is more important, e.g., to find a special type of earth's surface in satellite images.

1.1.2 Uses of Image Retrieval Systems

Image databases and image search engines are needed especially in geographic information systems (GIS), in interpretation of satellite images [KCH95], and in health-care systems for medical images [Kim, Gup95, G.R92, ZDS+], e.g. to detect cancer, in image archiving [Cat96], in agricultural (tele)diagnostic, in TV and News production [OSEMV95], in retail systems for selecting wallpaper, fabric, clothing [FSN+95], in on-line shopping, in scientific research for classification and building the taxonomy in botany, zoology; identification systems for security purposes; in face recognition [BPJ93, PPS94], in art libraries [Cat96], in copyright firms for finding similar, fake company logos. Incidentally, an image search engine on the Internet will help users who are currently overburdened with huge amounts of information returned while searching for a company, not by its (unknown) name, but by certain properties of its logo, for example.

1.2 Motivation and Research Objective

In databases, one of the commonly used access methods is indexing, because it permits relatively fast retrieval using one or more keys. The most popular indexing techniques for alphanumerical data are based on B^+-trees [Com79]. However, in multimedia databases, due to performance issues, conventional indexing techniques should be replaced by more suitable and efficient ones. Therefore multi-dimensional indexing structures are needed due to the multi-dimensional aspects of image features and image queries. There is relatively little research on indexing the multimedia data compared to variety of multimedia data types.

Most of the existing image retrieval systems, and commercial image databases either depend on the textual commercial databases [OS95], where image indexes are constructed according to the annotated keywords, or use the pioneering multidimensional index structure R-tree [Gut84], its variants and k-d-B trees [Rob81]. For example, R-tree and its variants are used in QBIC [FSN+95] and k-d-B trees are used in some parts of Chabot [OS95]. However, none of

the recently proposed multidimensional indexing structures such as TV-tree [LJF94], SS-tree [WJ96], X-tree [BKH96] is used in existing image retrieval systems.

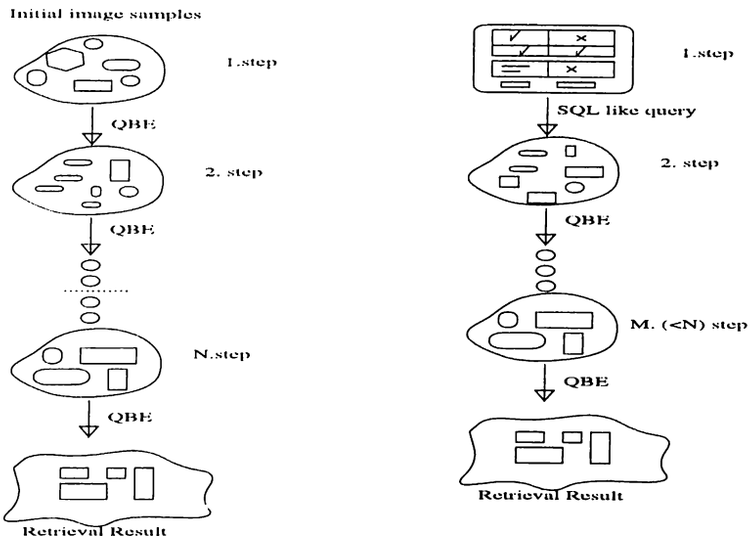

Figure 1.2: Comparison of only QBE and SQL+QBE type image retrieval systems

Another performance issue in image search engines is the support of the shortest path to reach the target image, which is only available by a combination of SQL (Structured Query Language) and QBE (Query By Example) (Figure 1.2). In the QBE type queries, a user is forced to select an image from the sample set. However the sample set may contain completely different images, e.g. images of nature or cars, while the user is searching for an apple. Therefore it takes some steps till s/he reaches the apple images.

Therefore in the system proposed here, a combination of IR and a new indexing structure, TV-tree [LJF94] is used, and the relevance feedback information of each retrieved image is provided to guide the user towards the target image(s).

Image signatures, that are actually bit vectors, are used as image representatives in content-based searches. Signatures describe not only the content as a whole, but also the objects residing in the image. Image signature calculation is explained in chapter 6.

Similar image signatures can be grouped together to build a group signature, in order to avoid comparing (bit-wise comparison) with all signatures during image query processing (e.g. two level signature files). Therefore it is highly optimized for speed. Because, instead of exhaustive comparison of signature files and query signature, query signatures are compared with only group signatures. In the prototype system, there is no need to employ the grouping mechanism because of the small database size.

In this thesis, signatures are not only used as a filtering mechanism for just the color feature as in the case of QBIC [FSN+95], but also for shape, texture and spatial features of images, which has not been used before, according to the literature survey.

Given an example feature vector (which is computed using the user presented sample image, like the computation of the signature) a list of similar matches can be retrieved from the constructed multi-dimensional index structure containing thousands of feature vectors in just a short time. A large number of images can be ranked very fast, a lot faster than by the use of more classical techniques of image content comparison, namely correlation.

There is a gap in the image segment indexing and image segment retrieval. Existing image retrieval systems employ the similarity based on color discrimination only for the whole image. However, some partitions of images should also be indexed, because those portions may contain some vital information which might be otherwise omitted. For example, a tumor in a brain tomography might occupy only a rather small part. However, it is the core information of the tomography image. Image segments are also indexed in the proposed system.

1.3 Overview of the Thesis

This thesis is organized as follows: Chapter 2 gives a survey of related work, highlighting the shortcomings of the existing technology. Chapter 3 explains the retrieval models and image query types employed in the image retrieval

systems. Chapter 4 presents the relevancy between Information Retrieval & Image Database Systems and the reasons for using signature files in image retrieval are explained. Chapter 5 explains multi-dimensional indexing problem and TV-tree structure with examples and proposed solutions. Chapter 6 presents details of the proposed solutions that are supported with algorithms and examples. Chapter 7 lists the conclusions and future works.

Chapter 2

Previous Work

There are two types of image databases: one with no image-understanding capability [OS95], and the other one with vision systems [PPS94], which stores images in a basic image repository.

The former one contains textual summaries for each image and indexes that use these annotations [CD95]. The lack of common vocabulary, inaccurate interpretation of images or insufficient keyword selection causes incorrect retrievals. Moreover, it is hard to do complex queries on images.

The latter is used for vision applications and research. However, there is less emphasis on specific database processes such as insertion, indexing, and querying. The common feature of both systems is the analysis and retrieval based on the actual image information content.

Image features representing the content can be extracted manually, semi-automatically, or automatically [FSN+95, KB96]. Each method has some advantages as well as disadvantages. Image features are multi-dimensional, i.e. there is no fixed query pattern for image retrieval and image queries, thus all image features should be indexed to support efficient querying. Moreover, query performance is more important than update performance in image database systems.

2.1 Multidimensional Indexing

The terms, *multi key*, *multi-attribute* and *multidimensional* are referred to access methods for secondary keys. In traditional database applications, image data are represented by records on disk which are N-dimensional points. For example, if $N = 1$ then the access is only by primary key and it is called one-dimensional (1-D). If $N > 1$ then it is called multidimensional.

The conventional indexing techniques based on B^+-trees are also not suitable for indexing the multi-dimensional data [NOL95], e.g. spatial data which occupy non-zero regions in the space. However, query performance is more important than update performance in image database systems. Therefore new indexing structures are needed to index visual data.

The research on indexing the multimedia data is very limited compared to the research activities on multimedia data types. Some indexing structures are proposed to support multi-dimensional data, such as k-d-B trees [Rob81], Grid Files [NHS84], Quad-tree [Gar82], R-tree [Gut84], R^+-tree [SRF87], R^*-tree [BKSS90], TV-tree [LJF94], SS-tree [WJ96], X-tree [BKH96].

The comparison of these indexing structures is given in chapter 5. However the presentation is kept more focused to some of them to explain the concepts used in the thesis.

2.1.1 R-tree

An R-tree [Gut84] is an index structure for point and spatial data at the same time. It is a height-balanced tree similar to a B-tree. Only leaf nodes contain pointers to actual data objects. The index structure is dynamic, balanced and no reorganization is required after deletion or insertion.

An R-tree uses a tuple to represent a spatial data in the database. To facilitate fast retrieval, each tuple has a unique identifier, which is lower and upper bounds of the bounding rectangle along dimensions, i.e. the internal nodes contain minimum bounding N-dimensional rectangles that cover the area

of all children nodes. Leaf nodes contain pointers to the actual data items.

Insertions and splitting of R-tree are made as in the B-tree. The deletion algorithm contains the call to the merging algorithm which condenses the tree structure. The condensation is made by re-insertion, because the conditions for merging nodes are different from the ones of the B-tree due to the multidimensional properties of the objects bounded by the minimum bounding regions. While searching, each time, the branch(es) intersecting with the query region or point should be followed. A search example of R-tree with 2-D data is given in chapter 5.

Originally the R-tree was proposed for indexing 2-dimensional spatial data. However, it is also used for similarity searching by image content in large image database. The R-tree suffers from the overlapping problem which is explained in chapter 5. Some variants of the R-tree are proposed in the literature to overcome the performance drawbacks.

2.1.2 R^+-tree

An R^+-tree [SRF87] is an extension of a k-d-B tree [Rob81] to cover non-zero size objects. Multi-dimensional space is partitioned into disjoint parts, in order to avoid overlapping rectangles in the intermediate nodes. Although region overlapping is avoided, a multiple path search is still needed, because each spatial data with non-zero area may be divided into different disjoint regions.

To search and to insert a data into the tree structure, R^+-tree exploits the same concepts of B-trees. The main difference is in the split propagation that is made upwards and downwards in the tree structure. The delete algorithm is the same as for the R-tree, but here, it is possible to delete several minimum bounding regions from leaf nodes because the insertion routine may introduce more than one copy for a newly inserted rectangle.

R^+-tree is proposed for supporting special applications of computer vision, CAD/CAM and robotics.

2.1.3 R^*-tree

An R^*-tree [BKSS90] tries to minimize the overlapping regions with its heuristic optimization algorithm. It incorporates a combined optimization of area, margin and overlap of each bounding rectangle in the internal nodes.

The overlap between bounding regions is decreased because of the effort to minimize bounding regions. Minimizing the overlap also decreases the number of paths to be traversed. The margin of a bounding region is tried to be minimized to make it more quadratic. For a fixed area, the object with the smallest margin is the square. So, quadratic rectangles can be packed easily and thus bounding a smaller rectangle.

It has the best performance among the other R-tree family members. However, all the R-tree based indexing structures will become a linked list, if the size of a single feature vector is bigger than a disk page. Moreover, the tree structure is dependent on the insertion order, i.e. tree structure is not unique. In some cases, the order of data is more apparent in performance of indexing structures.

2.1.4 TV-tree (Telescopic Vector Tree)

As stated in [WJ96] , a TV-tree [LJF94] is a unique structure which is actually designed for indexing the multi-dimensional data. The main idea of a TV-tree is to use a variable number of dimensions for indexing. The name of Telescopic Vector Tree stems from the contraction and extension aspect of feature vectors.

The number of dimensions depends on the data size to be indexed and to the level of the tree. In the first few levels, a small number of dimensions are used for indexing and a higher fanout is achieved, i.e. shallower tree. The discrimination is much more in the remaining tree levels with the introduction of new dimensions.

Performance of a TV-tree is better than that of R-tree based structures. The TV-tree structure and its algorithms are further explained in section 5.2.

2.1.5 SS-tree (Similarity Search Tree)

An SS-tree [WJ96] is based on the R-tree structure. The insertion algorithm is similar to that of an R^*-tree. However, an SS-tree uses spherical bounding regions rather than rectangular regions. An SS-tree relies heavily on a domain expert to help in the indexing process. The domain expert should specify groups which contain feature vector elements with the same weight.

The similarity distance measure is actually a weighted Euclidean distance metric. Each SS-tree structure uses one base distance metric. Thus its drawback is apparent while searching for data with different feature weights. More than one SS-tree with different base distance metrics are needed for indexing, because a single SS-tree can only be optimal for one base distance metric.

There are no test results available for approximate queries which are frequently used in content-based querying of multimedia data.

2.1.6 X-tree (eXtended node Tree)

The nodes of an X-tree [BKH96] are a hybrid of linear array and R-tree nodes. An X-tree uses supernodes to avoid degeneration in the indexing structure, i.e. supernodes are used for avoiding splits of overflowing nodes which may cause new overlapping regions. A supernode is a linear array like structure and its size is variable. Usually its size is twice the block size. Actually an X-tree contains three different types of nodes: data nodes, normal directory nodes (internal nodes) and supernodes.

The proposed split algorithm uses the track of previous splits for finding the optimal split. The optimal split is actually the split which does not cause any new overlapping region. Point, range and nearest neighbor algorithms are similar to R^*-tree algorithms, except the supernode concept. An X-tree performs better in insertion and retrieval than R^*-tree and TV-tree.

The major problem of multi-dimensional indexing structures is the overlapping region problem (explained in chapter 5) caused by the growing dimension.

2.2 Image Retrieval Systems

Lycos[1], Alta Vista[2], Yahoo[3], Excite[4] and other Internet search services index a significant portion of the World Wide Web by their textual content and make searching available to the public. For example, Lycos extracts keywords from documents using the word placements or their frequencies. They are designed for indexing a huge number of Web sites and use a "*Boolean*" search method in which the user can only search for exact words or parts of words or combinations of words that exist in web pages and they are based on alphanumeric data that is inadequate to represent visual data.

Chabot[5] [OS95], QBIC[6] [FSN+95], WebSeek [SC97a], VisualSEEk[7] [SC96], WebSeer[8] [SFV], Safe [SC97b] are designed for searching visual data that is available via the Internet. They enable to query either images residing in their local database or images available in the Internet.

2.2.1 Image Search Engines for the Internet

WebSeek[9] [SC97a] offers content-based relevance feedback, WebSeer [SFV] tries to combine image content information with associated text, however their user interfaces permit only textual queries. VisualSEEk [SC96] and Safe [SC97b] are the latest Java enabled search engines, however they do not exploit the power of signatures, such as less storage need and fast matching processing (bitwise comparison). Apart from these image retrieval systems, some image engines [FSN+95, OS95, PPS94] are available only on local image databases.

[1] http://www.lycos.com
[2] http://www.altavista.com
[3] http://www.yahoo.com
[4] http://www.excite.com
[5] http://elib.cs.berkeley.edu/cypress/
[6] http://wwwqbic.almaden.ibm.com/ qbic/qbic.html
[7] http://disney.ctr.columbia.edu:8021/VisualSEEk/VisualSEEk.html
[8] http://infolab.cs.uchicago.edu/webseer/
[9] http://disney.ctr.columbia.edu/WebSEEk/

2.2.2 Image Retrieval Systems in the Internet

QBIC by IBM

IBM's Query By Image Content (QBIC) [FSN+95] was the first complete system to demonstrate the use of simple attributes in appearance-based retrieval of images from a reasonably sized database. In QBIC [FSN+95], instead of time consuming computations, first a filtering mechanism is employed to reduce the search space only for color histogram matching, then a complete matching operation is applied to the resulting candidates [FBF+94]. Color histogram matching is explained in 6.4.2.

However, the highest retrieval rate is in color and its 10% success rate [Cat96] is very low (i.e. 90% of the images in the resulting set are *false drops*). The filtering operation, 256-dimensional color matching, consumes a lot of time especially in image databases with thousands of images. Another drawback of this filtering arises where images have similar color histograms, but different shape or texture information. Although images satisfy the filtering condition, they will fail in fine tuning. The use of shape indexing in QBIC is also problematic when manually creating shape information [Jai96].

QBIC is used for the content-based indexing and retrieval of image collections from French Ministry of Culture and Fine Arts Museums of San Francisco [Cat96, FSN+95].

Chabot by UC Berkeley

The UC Berkeley's Chabot [OS95] project annotates the images with textual data that are stored in an object-relational DBMS called Illustra which is a commercial version of UC Berkeley's POSTGRES research project [Gro93].

All the image content and feature information, including the color histogram information are stored as text in the database, where no similarity searching is available. Drawbacks of keyword annotation are explained in section 6.1. "Exact match" queries also fail to match images with small deviations. Support

for integration of image features with text and other data types is essential. Again B- or R-trees are used as index structures for user-defined indices with their inherited performance drawbacks [LJF94, WJ96, BKH96].

Photobook by MIT

The Photobook project [PPS94] at the MIT Media Lab uses neither keywords, nor textual transformations of image features for indexing images. It actually stores the encoded image segments themselves which can be reconstructed for direct search on image content. It applies semantic-preserving image compression, i.e. compact representations that preserve essential image similarities.

Image segments are transformed into a coordinate system that preserves perceptual similarities, and then uses a lossy compression method to extract and decode the most important parts of that representation. The approach, "perceptually complete" and "semantically meaningful" image encodings, is the pioneering approach for content-based retrieval.

Photobook system stores also textual annotation information in an object-oriented, memory-based AI database called Framer [Haa93]

It provides color histogram computation at run-time. Although a lesser amount of information is stored, a much longer time is consumed for reconstruction, uncompression and other post-processes in image retrieval.

2.2.3 Other Image Retrieval Systems

In the PICTION system [Sri95], the textual captions of newspaper photographs are processed by some NLP techniques to help predict and identify objects in images. The descriptive text can only explain the main content of the image but not each of its minor points. For example, satellite images or medical images are best examples to degrade the PICTION system's effectiveness. In cases where an image is worth a thousand words, keyword annotation or assignment, and clustering into categories can not solve the content indexing problem.

The basis of the CANDID (Comparison Algorithm for Navigating Digital Image Databases) [KCH95] approach is to describe either an entire image or a specific region of interest with a global signature of texture, shape or color content computed by the *N-gram* method [KCH95]. The probability density function is the content signature for the given image. However, no fine tuning or textual search is supported in CANDID.

2.3 Signature Approach in Image Retrieval

In QBIC [FSN+95, FBF+94], color signatures are used for filtering the color information and then a second phase is employed for eliminating the *false drops* (explained in chapter 4). However, there is no filtering phase for other image features, like shape and texture.

In CANDID [KCH95], image features (local color, texture and/or shape) are extracted for signature computation. Image signatures are computed by making use of the probability density functions.

Chang, et.al [CSY87] have developed an indexing mechanism which uses character strings to represent spatial relationships between different objects in an image. Due to the space and performance limitations of this scheme, spatial information of image objects are converted to spatial signatures [LYC92]. These signature are intended to accelerate the matching process, where normal string matching algorithms were employed before. Two-level signature files are adopted for the filtering phase.

Using spatial signatures may cause some problems, since for spatial relationships, the relative ordering of the specified image objects should be reflected to the signatures which are also convertible easily to the inverse of spatial relationships.

Therefore the stored signature and the query signature should be processed somehow to facilitate the retrieval.

Chapter 3

Retrieval Models in Image Databases

Searching text just requires a query on algorithm-based extraction of keywords, which does not need an understanding of the text meaning. The text content is tried to be captured by some keyword extraction algorithms. The keywords are considered as the content descriptors of the documents. For example, the keywords of the textual documents are selected from the ones with the highest word frequencies or from the distinct words. Or word locations are considered in keyword selection, e.g., the word that are after the *stop words* are selected as keywords.

The selected keywords and the document to be processed are in the same type (i.e. both are alphanumerical data), hence it is easy to describe the content of the document with the keywords.

However, it is not the case in visual documents. Image features and the occasion captured in the image actually defines the content. Therefore visual feature extraction algorithms are needed, which is a more overwhelming task than keyword extraction. Pattern recognition and computer vision techniques are needed for fully automatic feature extraction. Moreover, similarity with respect to human perception sometimes differs from the algorithmic computation. For example, humans perceive that shapes are similar even when they

are deformed versions of each other. Almost none of the existing algorithms for finding the similarities perform well when there is non-rigid deformation.

Searching images in a database mainly depends on three major comparison methods: Boolean, vector space and probabilistic retrieval models [KB96]. The first of these is based on the "*exact match*" principle which is employed in standard textual databases, the other two are based on the concept of "*best match*", which are employed in multimedia systems, where the non-alphanumerical content has more expressive power than the keywords.

3.1 Boolean Retrieval Model

The annotation of meta-data, keywords and captions of images can be adapted to content-based queries, because content-based retrieval depends on the richness of the meta-data or meta-knowledge about the image itself. Textual information may describe the image by semantic information (imaging date), image attributes, image object properties (type of shape), domain-defined objects (tumor in brain tomography), measurements of domain or image objects (size, area) and their absolute or relative positions.

Normally the annotation should be done automatically by the system with the help of a domain expert during the insertion to the database. However, today none of the existing systems, such as QBIC [FSN+95] and Chabot [OS95] can do it correctly. Image objects can not be recognized fully and correctly. There is always some noisy, deficient or incorrect keywords attached to the images, or some image information is lost due to the expressive power of natural or formal language.

In the Boolean retrieval model, images are indexed with the assigned keywords and they are retrieved by either SQL or free-text matching. Although it is easy to specify abstract queries, the retrieval performance is as good as specified text descriptors.

Other drawbacks of keyword based indexing are explained in chapter 6.1.

3.2 Vector Space Retrieval Model

In vector space model, image descriptors (feature vectors) are computed according to a pre-determined feature set. Due to the size of image information, generally their descriptors are used in image searches. Image descriptors are actually feature vectors. A feature vector is a bit vector where a bit is set (i.e. "1") if a feature is known to be true, and not set (i.e. "0") otherwise. The key characteristics of an image are distilled and its feature vector is created, which is actually the signature of the image. Due to compactness and fast comparison (bit-wise comparison) image signatures are preferred to associated text or keywords for content-based retrieval.

For example, let there be N predefined image features ($f_j, 1 \leq j \leq N$), and v_i is the feature vector of image i,

$$v_i = (f_1, f_2, ..., f_N)$$

If f_j exists or is satisfied by the image i, then it is set to a specific value, e.g. "1", where non-existent features are denoted by "0".

The similarity distance of two images is actually calculated by the "furnished" Euclidean distance of two vectors, "furnished" means that some domain knowledge should be embedded into the similarity computation. For example, the similarity distance of yellow to orange is less than that of to red. This perceptual similarity should be reflected to similarity calculations.

3.3 Fuzzy (Probabilistic–Semantic) Retrieval Model

In this retrieval model, past experiences or statistics based on many sample queries, help improve the retrieval performance of new queries. Conceptual categorizations and concept maps are used to indicate the relationships between objects [LOS95]. The features of images are restructured into semantic

networks to represent knowledge in an interconnected manner. The precise definitions of image elements and their proximal relationship to one another are also exploited in fuzzy (semantic) based queries. For example, an image categorized to the topic of "sports" taken in light-weight championship is unlikely to be retrieved while searching for the image of a Compact Disc, although both images contain circular objects.

In a fuzzy query a user may fire a query like; "Give the *most impressive* pictures of Bosnia showing the *bad impact* of war".

A fuzzy retrieval model exploits multidisciplinary techniques, such as Artificial Intelligence, Machine Learning, Information Retrieval and Natural Language Processing. Therefore its accuracy is limited with the problems in these multi-disciplinary areas.

Fuzzy searches or "similar to" queries are used in multimedia databases, since words do not have the same expressive power as that of an image, or sound. Fuzzy or probabilistic queries for images make use of Information Retrieval (IR) and Artificial Intelligence (AI) techniques in database systems. As it is stated in [KB96], image databases are the combination of database capabilities and information retrieval capabilities.

Comparison of Retrieval Models.

Retrieval Model	**Advantages**	**Disadvantages**
Boolean	Easy complex query specification	Vocabulary dependent No similarity queries Larger index size
Vector Space	Adaptable and portable Easy to implement Similarity retrieval is supported	Dimension problem Need for good similarity function
Fuzzy	Concept-based queries Easy semantic query specification	Domain-specific No perfect algorithm Multidisciplinary limits

3.4 Query Types in Image Databases

Query-By-Example (QBE) type of queries are common in image search engines [FSN+95], which helps the user navigate through an iterative refinement procedure towards the goal. However, it sometimes makes getting closer to the goal almost impossible, for example, representative images selected from the database may be unrelated with the query image. Therefore, a combination of QBE and SQL like visual languages should be applied for successful searches and to cut down the search time.

There exist some systems, such as QBIC [FSN+95], in which the user either draws a rough outline of the image or a template for querying. This is called as "*Like this retrieval*". However, it is very hard to draw a template if the query object is very complex to draw, for example tumors in brain. Thus QBE is frequently used in image retrieval systems.

Another type of QBE is the iconic query. The image icon gives a rough idea about the real image. The representative objects, color or textures of the image is selected for its iconic representation.

QBE type queries are employed in *vector space* and *fuzzy* retrieval models, but *Boolean retrieval model* is especially used in SQL like queries.

Chapter 4

Information Retrieval & Database Systems

In the past, conventional Database Management Systems (DBMSs) have typically managed simple data types such as strings and integers. One of the current trends is the use of DBMS for the management of multimedia data, particularly as software, and computers become better able to handle audio and video data requirements. Certainly, multimedia data has been stored in DBMS since 1980's, but with severely limited support. Object-Oriented Database Management Systems (OODBMS) (vs. relational DBMS) have generally become the database management system of choice for multimedia data, which is better supported by OODBMS [RNL95].

In conventional DBMSs, the "exact match" principle is applied. The result of a query is a single set, whose members actually validate the query term, no false information exists in this set and none of the qualifying information is ignored. IR systems also return a single set, whose members are ranked since the retrieval involves content queries that include weights. The order of results indicates the closeness of the query term with the result. However, the result set may contain false data, e.g. false drops.

Fuzzy searches and iterative queries (for refinement at each step) are needed in image databases, because features extracted to index image data or search

the database are difficult to identify and not precise enough. As in textual IR systems, similar techniques can be applied to rank the results in image retrieval.

Probabilistic or Boolean comparisons are also used in image searches. A major problem with the Boolean model (used in database systems) is that it does not allow for any form of relevance ranking of the retrieved data set. In the "best-match" retrieval method, terms can be weighted according to their importance, which are computed on the basis of statistical distributions of features in images. In probabilistic comparison, retrieved information is ranked in order of probability of relevance to the query, given all the evidence available. The most typical source of such evidence is the statistical distribution of features in the database, and in relevant and irrelevant images.

Traditional databases are the abstraction of the real world pertinent to the problem at hand in terms of alphanumeric data. However, image databases are a close marriage of the three fields; database management, information retrieval and computer vision, each of which needs to make some adjustments. Storing multimedia data in MMDBs has several uses [PS96].

4.1 Signature Files

Although signature files are first used in IR for content-based searching of textual information, they can also be used in searching multimedia data [KCH95]. The signature file approach is one of the most powerful methods for information retrieval [FC84]. The main idea is to convert the given information into a bit sequence by using a hash function, which is then stored in compressed or uncompressed form into separate files. Each representative or important part of information is converted to a bit stream and combined to build the signature of that information. For example, suppose that "red", "car" and "collision" are keywords of the document D_i. In order to create the signature of D_i, each keyword is converted by a hash function, h_i into the following bit streams:

Keyword	Signature
red	0000 0101
car	1000 0010
collision	0100 0010
D_i^{sc}	1100 0111
D_i^{ws}	0000 0101 1000 0010 0100 0010

The document signature is created either by superimposing all the keyword signatures, as in the case of D_i^{sc} (Super imposed Coding Method-SC [FC84]), or by concatenating them one after another, as in the case of D_i^{ws} (Word Signature Method-WS [FC84]). The former suffers from *false drop*s and the latter suffers from *space inefficiency*, because more bits are used for the signature. The SC method is the one generally used in document signature computation.

Instead of string matching, bit conjunction (bitwise *AND*'ing) between query signature and document signature reveals the result of the query. Although bitwise comparison is much faster than string matching, the probability of *false drop*s still exists. A *false drop* is the mapping of different words into same bit positions. Some qualified documents seem to contain a keyword, which is, in fact, non-existent. For example, the signature of "sum" computed by h_i is "1100 0000". Although the document signature D_i^{sc} satisfies it, it does not exist in D_i. Therefore the hashing function h_i to generate the signature can be tuned in such a way that the *false drop*s can be adjusted within tolerance. To overcome the *false drop* problem some solutions are proposed [AC93].

4.1.1 Signature Files in Image Databases

Signature files are used for searching image databases, where images are manually annotated with keywords [Ce86]. However, recalling the old cliche, it makes the signature files space inefficient. So, a new set of principles need to be established for image analysis and representation, since the assumptions on text signatures are inapplicable for multimedia data, e.g. the probabilistic independence of word occurency assumptions are not valid for image databases, because there exist some semantic rules that construct complex objects from the primitive ones [AC93].

An image signature is a compact form of some feature information of the image such as color, shape, texture which can characterize its content. An image can have more than one signature associated with it, e.g. for each predefined image segment, because the content of visual information is well suited for variable and subjective interpretation.

4.1.2 Advantages of Using Signature Files vs Other Structures

The high storage need of image information itself and its interpretation causes some problems in efficient data storage and data retrieval while designing an image database system. When retrieving images from the database, some discrimination terms are defined. Images, actually image information extracted from images, are checked with the discrimination terms and satisfying images are returned. The checking of the encoded image information should be done very fast and the space reserved for that information should also be minimum to be efficient. Therefore the signature file approach is an excellent choice for using as a filtering mechanism in image searching.

Signature files require very little processor time during the resolution of conjunctive queries, especially, in machines where a special hardware is built-in for performing bitwise *AND* operations. This property makes signature files more preferable than inverted files, due to the latter's inefficient performance in conjunctive queries [WMB94].

In some cases, not all the pixel values or objects in the image are important, due to the limited capability of human perception or errors in image creation tools, e.g., low resolution of scanners. This situation is analogous to "full-text" searching and "keyword searching" in textual IR systems, where inverted files are preferred in the former, and signature files in the latter due to performance reasons [WMB94].

However, the signature file approach does not provide good ranking support, and for applications requiring feature's spatial information signature files need to be augmented by other structures.

Chapter 5

Multi-dimensional Indexing

Index in a database is actually an entry for each data item, that contains the value of the key attribute for that data item and a reference pointer that facilitates immediate access to the location of the data item. The most popular indexing techniques for alphanumerical data are based on B-trees [BM72] and their various extensions such as B^+-trees [Com79]. However in multimedia databases, due to performance issues, conventional indexing techniques should be replaced by more effective and efficient ones [NOL95].

In content-based queries, all images are represented by some pre-computed visual features. The key attribute of an image will be a feature vector which is represented as a point in a multi-dimensional feature space. Similarity queries depend on the similarities between feature vectors. Therefore an efficient multi-dimensional indexing scheme is required in order to achieve fast and effective retrieval in content-based queries.

Color, shape, texture features and their spatial positions are the common and mostly researched attributes of visual data, because of their distinguishing capabilities. However, each of them has many subfeatures. For example, color can be any of the 16 million colors; shape can have color, formation or spatial orientation features; texture features can be determined according to edge density, randomness, orientation; spatial positions in 2-D space can be represented in 169 different cases [LYC92].

All these characteristic may be needed in image retrieval, because there does not exist a fixed query pattern in content-based queries. In some applications, shape and color are rather important than texture and spatial features, but not in others. Therefore images should be indexed in a way that they are available for searching in a reasonable time, regardless of the number of images.

Therefore image content representation is converted into a set of coordinates for a point in the multi-dimensional space and then any multi-dimensional index structure, such as *k-d-B trees* [Rob81], *linear quadtrees* [Gar82], *grid-files* [NHS84], R-tree [Gut84] and its variants R^+-tree [SRF87], R^*-tree [BKSS90], TV-tree [LJF94], SS-tree [WJ96] and X-tree [BKH96], can be employed for indexing.

5.1 Idea behind the Multi-dimensional Indexing

There are two different approaches on indexing multi-dimensional data;

- The observations on the real data reveal that most of data points in multi-dimensional space are highly correlated. They occupy only a subspace with a lower number of dimensions. Thus some transformations (e.g. FastMap [CF95], Karhuenen Loeve) can be used to lower the number of dimensions of space [NOL95], so that they can be indexed using traditional multi-dimensional structures.

- The observations on the most of the high-dimensional data reveal that a small number of dimensions contains most of the information, i.e. a small number of dimensions has more importance than others during querying.

Fast retrieval and fast access to multi-dimensional data is only possible by decreasing the search space. The similar data points are grouped in such a

way that a minimum bounding region contains them. The minimum bounding region is actually the representative of the set of points in that region. The minimum bounding regions are then inserted in a tree like structure.

Each node contains the minimum bounding region of its child nodes and the leaves contain pointers to the actual objects. Generally, the nodes are considered as disk pages, thus, the parameters (page size, fanout, fill rate, percentage of overlapping region etc.) of the tree structure should be chosen carefully to accelerate the search on the tree structure. During the visit of each node, a disk page read time is spent. For example, total search time is about 32 seconds in an R-tree with 100 MBytes of entries with 10 dimensional real data [BKH96] on an HP735 workstation with 64 MBytes of main memory.

The main problem of multi-dimensional indexing:

> *The higher the dimension of the information, the more degradation in query performance on index structure. The number of overlapping regions increases with the increasing dimensions. Therefore multiple branches of the tree structure are traversed unnecessarily.*

The overlap of minimum bounding regions, i.e. directory nodes, is directly correlated to the retrieval performance. Multiple paths should be traversed even for point data which intersects with the overlapped region. As the research indicates that the number of overlapping regions increases with the growing number of dimensions [LJF94, BKH96, WJ96].

In the proposed multi-dimensional tree structures, bounding regions of different nodes may overlap [Gut84, SRF87, WJ96, LJF94, BKH96] or an object may be located in several nodes [SRF87]. Both methods have some drawbacks. The former suffers from overlapping regions. While searching an object in the tree several nodes should be visited prior to be sure on the presence or absence of that object. The region overlapping problem is solved in the latter by partitioning the space into disjoint regions.

However, while searching for all objects in a particular region, many of the objects are retrieved more than once, i.e. segments of the same object in

different regions. Still multiple branches of the tree should be traversed, which becomes similar to a sequential algorithm.

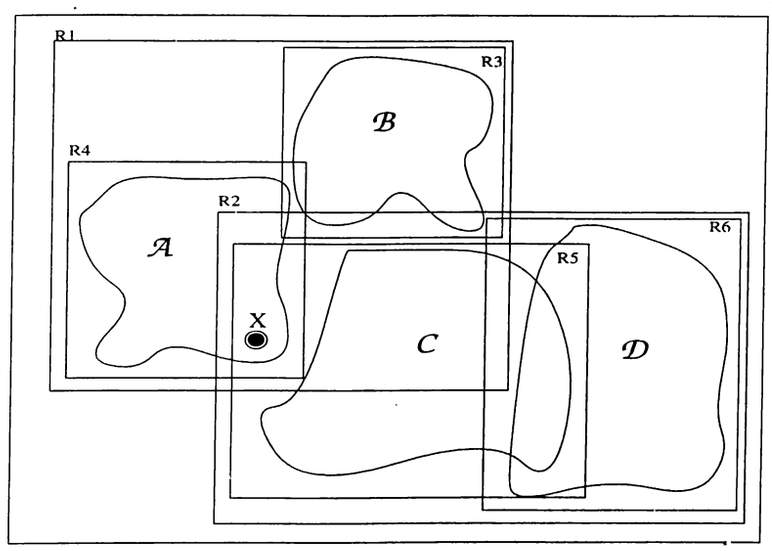

Figure 5.1: 2-D Data Sample

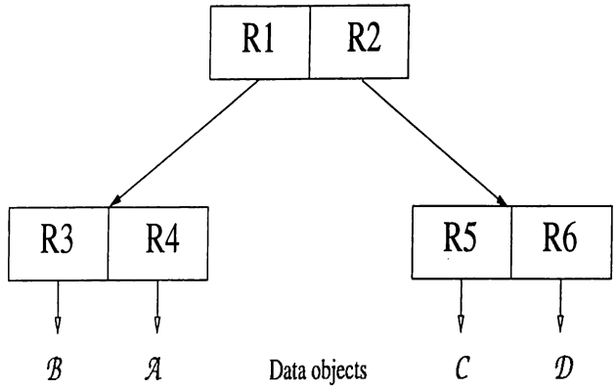

Figure 5.2: Tree Representation of 2-D data

An Example

The Figure 5.1 is an example for the overlapping problem in the multidimensional space. For convenience only 2-D spatial data are taken. The problem is to retrieve the object which intersects with the given search point X. The amorphous data objects A, B, C and D occupy some area in 2-D data space. The minimum bounding rectangles $R1$, $R2$, $R3$, $R4$, $R5$ and $R6$ partition the space into subspaces. For example, $R3$ is the minimum bounding region for

object B. The R-tree representation is shown in figure 5.2. While searching the point X in the tree structure starting from the root, both branches should be searched, since either $R1$ or $R2$ may contain X. Searching $R1$ leads to the bounding region $R4$ where the point X is actually in it. So we succeed. However, the right branch of the root should also be searched, because X also intersects with the region $R2$. At this time, the region $R5$ intersects with the point X, but not the object C. Therefore the right branch is time-consumingly searched.

These indexing schemes employ some partitioning methods to partition the multi-dimensional space. The main goal is to visit the minimum number of partitions, while searching for an exact query point or similar points (i.e. nearest neighbors) to a given point. However, this technique is not better than a sequential search, since performance degrades geometrically as the number of dimensions increases. For example, the directory in grid-files grows geometrically with the dimensionality. The performance drawbacks of these indexing methods are investigated in the literature [LJF94, WJ96, BKH96].

In this thesis, TV-trees which have not been used in any existing image search engines or image databases, have been employed and justified.

5.2 Telescopic Vector Tree - (TV-tree)

As stated in [WJ96], a TV-tree is the unique structure which is actually designed for indexing the multi-dimensional data. The main idea of a TV-tree is to use a variable number of dimensions for indexing. The name of Telescopic Vector tree stems from the contraction and extension aspect of feature vectors.

The number of dimensions depends on the data size to be indexed and to the level of the tree. For the nodes close to the the root, a small number of dimensions is used for indexing. So that a higher fanout is achieved (more entries are in the same node), i.e. a shallower tree. The discrimination is much more in the descending tree levels with the introduction of new dimensions. For example, let there be a tree with a fan-out of k and l levels, it can only

keep $O(k^l)$ number of data items. Thus when k increases, the tree capacity increases, and thus the tree can become shallower. Once the tree gets shallower, a few number accesses are required for searching.

There are two main assumptions on the TV-tree:

- The dimensions of feature vectors are sorted by their importance. The dimensions with higher weight become the first few dimensions for comparison.

- Each feature vector should have the same number of important dimensions of the same type.

In a TV-tree, the data is organized in a hierarchical structure; feature vectors are clustered in the leaf nodes of the tree, and the description of their minimum bounding region is stored in its parent node.

The TV-tree is chosen as the indexing structure in this work, because its performance is better than R-tree like structures for higher dimensions, e.g. dimensions greater than 16 [BKH96].

Although a TV-tree has a better performance than the R-tree family, there are some critisizms on the TV-tree indexing structure. In [BKH96], the authors argue about the maximum number of data items that a TV-tree can support. The second assumption of a TV-tree is claimed to be unrealistic in [WJ96]. In the conclusion part of this chapter, a solution is proposed to overcome these problems.

In the following sections, the structure, insertion and search algorithms are stated briefly. The details of the TV-tree can be found in [LJF94].

p	Distance metric
N	Number of dimensions
c	N-dimensional center vector of the sphere
r	Radius
a	Number of active dimensions
x	Set of points bounded by the spherical region

Table 5.1: TV-tree node parameters

5.2.1 TV-tree Structure

Each node of a TV-tree represents the minimum bounding region of all its children like in R-tree. However the shape of the bounding region is a sphere. Each region is denoted by a center, which is a vector (coordinates of the center) and a scalar radius.

The shape of the minimum bounding region that is simplest to represent is the sphere. A sphere of radius r contains the set of points with Euclidean distance (special case of L_p-metric, where $p = 2$) $\leq r$ from the center of the sphere. Euclidean distance is actually the physical distance between two points in the space.

General distance metric: $L_p(\vec{x}, \vec{y}) = [\sum_i (x_i - y_i)^p]^{1/p}$
Euclidean distance: $L_2(\vec{x}, \vec{y}) = [\sum_i (x_i - y_i)^2]^{1/2}$

The sphere is *telescopic*, because it contracts and extends depending on the objects stored within the region. It is represented as follows using the parameters in Table5.1;

$$c_i = x_i \qquad 1 \leq i \leq N - a$$

$$r_p \geq \sum_{i=N-a+1}^{N} (c_i - x_i)^p$$

Active dimensions are the dimensions which have been taken into account while specifying minimum bounding regions. Inactive dimensions can not distinguish between the node's descendents in the tree structure. For example,

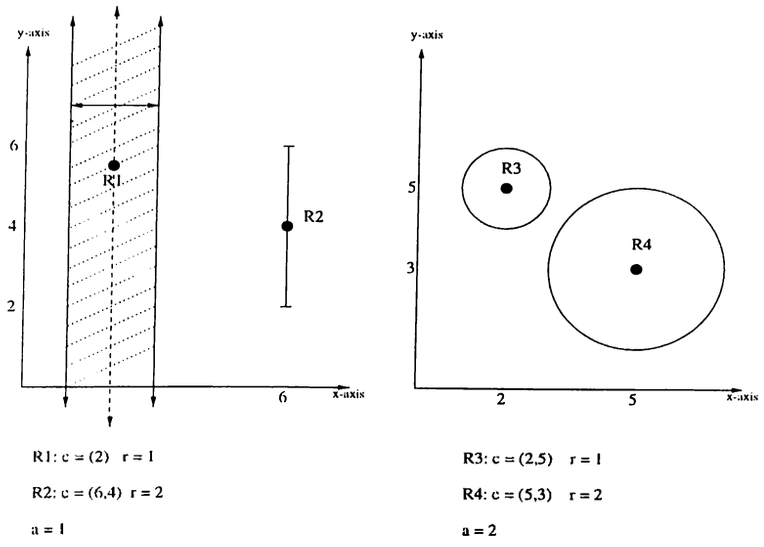

Figure 5.3: Example minimum bounding regions

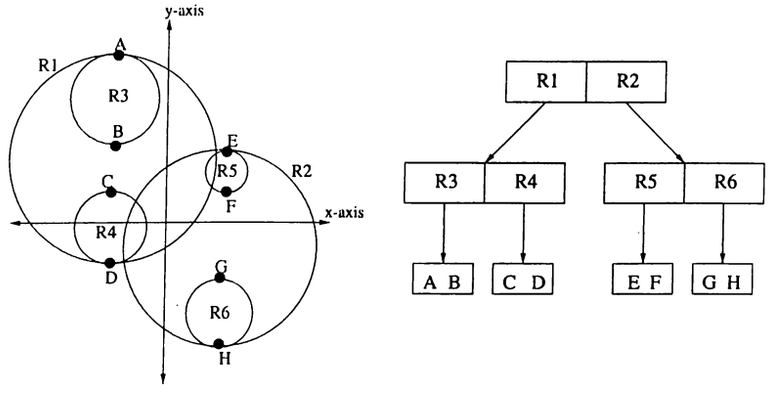

Figure 5.4: Example of a TV-tree with minimum bounding spheres

in Figure 5.3 R1 has one active dimension (a=1 and it is x-axis). The dimensionality of R1 is 1 (only the data along x-axis is used to specify R1 in the tree structure). The dimensionality of R2 is 2 (x and y axis are both used). R2 has 1 inactive dimension (x-axis) and 1 active dimension (y-axis). R3 and R4 are self-explanatory examples.

An example of the TV-tree structure with some data is given in Figure 5.4.

5.2.2 Insertion

Similar to the insertion algorithms in binary trees, the algorithm starts with the root and the appropriate branch is selected at each tree level. The traversal continues till we reach to a leaf, where the new data should be inserted.

In case of an overflow, either the node should be split, or its contents should be reinserted; *i.* to minimize the overlapping regions, *ii.* to arrange new bounding region values, and *iii.* to avoid deterioration in the tree structure.

At the end of insert/split/reinsert operations, some internal nodes are affected, i.e. the value of the bounding region might have been changed. Thus, the values of all affected nodes should also be updated.

There are two types of update, either the radius of a bounding region may increase or the number of active dimensions may decrease [LJF94].

5.2.3 Searching

As in the insertion algorithm, the tree is traversed starting from root down to the leaf level while examining all the branches, which intersect with the given search region. The search region is actually a vector.

The multi-dimensional problem still exists in a TV-tree for cases where a given data object intersects with different bounding regions, or where minimum bounding regions intersect with each other.

5.3 Conclusion and Proposed Solution

The main problem of the multi-dimensional indexing methods is the deterioration of performance with the increasing dimensionality. Overlapping regions in the multi-dimensional space unnecessarily cause multiple path searches. Thus the retrieval on the index structure may turn into a sequential search.

To overcome this difficulty I propose to apply a filtering mechanism, image signatures, to reduce the number of dimensions. The filtering mechanism is applied before searching in the index tree. In the filtering step, not only the number of dimensions is reduced, but also the number of images to check is decreased. Filtered data are used as the actual entries of the index structure.

Thus the index structure will contain fewer number of entries than before. In [BKH96], the authors stated that they could not insert more than 50 MByte of data. This drawback of the TV-tree is solved by reducing the search space using signature filtering. Actually the TV-tree is used for fine-tuning and for detecting the false drops caused by the first stage.

Index Structure	Advantage	Disadvantage
Grid files	Guaranteed access to any record in 2 disk I/O, one for each level, one for grid	Not scalable for N (>2) dimensional data Transformation is required for spatial data The directory does not grow at the same rate as the data
k-d-B Tree	Useful only for 1-D data	Updating by reinsertion causes possible deadlock Very slow
Quad Tree	Better for non-uniform distributed data Good for set-theoretic operations	Difficult feature based queries, because of it is lack of feature indexing capability Space requirement is dependent on the position of origin (shift sensitive) Bad performance for uniform data
R-tree	Each object only in one leaf Suitable for both point & spatial data	Degraded performance in increasing dimension Overlapping regions Space inefficiency for indexes in higher dimensions Not unique, tree structure depends on the insertion order Expected B-tree performance is not guaranteed
R^+-tree	Avoid overlapping regions	Partitions of an object is in different leaves Not unique, dependent on data order Storage utilization may deteriorate Traversal of unpredictable length during insertion & deletion
R^*-tree	Minimize overlapping space Minimize dead space for making Decision in higher level so better performance than R^+-&R^*-tree	Insertion is quadratic Not effective for dimensions > 5, because about 90% of the bounding regions overlap
SS-tree	Similarity measures are actually stored Better performance than R^*-tree Insertion is linear	Multiple SS-tree's are needed for each query, if different similarity weights are used Excess domain knowledge is needed
X-tree	Minimize overlapping regions with supernodes Better performance than R^*-tree and TV-tree	Sequential search in large supernodes. Additional operations should be performed, in case of underflow in supernodes
TV-tree	Better performance than R-tree family	Still overlapping problem

Table 5.2: Advantages and Disadvantages of multi-dimensional indexing structures

Chapter 6

Image Search by Signature Filtering–SIS

6.1 Foundations of SIS

Textual information is composed of character strings that are direct representatives of the information. However, visual information (raster image data) contains only the color information of individual pixels, file name, size and other format data. None of the existing standardized image formats contain any content-related information. Therefore the extraction of image content information needs additional tools and techniques. Although the color information of individual pixels do not help with the content of image, their combination collectively can define a shape or pattern, which may describe the core content of the image.

A user should be able to search an image database for images that convey the desired information, e.g. a doctor should be able to search a corpus of diseases for images that are relevant to the patient's X-Ray within a couple of seconds [G.R92, ZDS+].

Most commercial image retrieval systems associate keywords or text with each image and require the user to enter a keyword or textual description of the desired image, because; i. there is no perfect algorithm or tool that perfectly

interprets the image content. **ii.** it is a straightforward process, and there is no need for employing complex pattern recognition algorithms.

However, keyword annotation has numerous drawbacks during indexing and querying phases, such as;

- annotating images with keywords or text is a boring task
- keywords may be under-descriptive for some image feature
- keywords are conducive to subjective interpretation, human intensive
- some features can be described in widely different ways, a standardized keyword dictionary is needed
- impossibility to define all objects in the image due to the space limitations

Proposed Solution

There are some efforts to overcome these problems and to improve retrieval accuracy. Researchers have mainly focused on content-based image retrieval in which retrieval is accomplished by comparing image features directly rather than textual descriptions of the image features. However, both image features and textual descriptions should be supported in image retrieval systems.

6.1.1 Image Query Language

Querying the images by a textual query language like SQL limits the expressive power of visual queries. The problem is that SQL is basically a means to retrieve tabular representation, however image features mostly require retrieval from a visual representation. Thus the SQL grammar should be extended by adding a set of tags or operators. Actually, the standard triple "*S F W*" (i.e. select...from...where...) syntax should be extended. However it is very difficult to generate a generic syntax, because there is no common and accurate query

syntax for image queries. Only spatial data rather than all image features are taken into consideration in the standardization effort of *SQL-3/MM* [SM].

The similarity measures have not been added into SQL like queries, because only Boolean operators are used for comparisons.

The inefficiency of SQL like queries in content-based retrieval reveals the need for efficient *Query By Example* (QBE) retrieval systems. For example, the user might provide a sample image and request other images;

- that are similar to the given sample,
- that contains similar objects with different colors, or vice versa.

Drawbacks of only QBE-type Queries

Many current image retrieval systems are based on appearance matching, in which, for example, several sample images are presented, and the user selects image(s) with similar color, shape or texture. However, this type of query may be time-consuming or unsatisfying for several reasons:

- It is mostly problematic to find a similar image from the presented images, e.g. searching an apple image by selecting an image from among images of flowers, animals, cars, buildings.
- Generally, it is hard to understand why particular images are returned
- Users have difficulty controlling the retrieval behavior in desired ways
- No relevance feedback to the user
- It introduces "semantic gap" between user's conceptualization of a query and the query that is actually specified to the system.
- No way to define the importance of image features in the retrieved images

Moreover, new query interfaces are required to support content-based queries, which provide examples of multimedia data to match. The user can then use

that example to find other examples in the database like it. This approach is called Query-By-Example (QBE). Image retrieval systems commonly use QBE, by which the users can draw an example of the image they want to retrieve, using shape, color or texture for example [PS96].

Proposed Solution

Most of the existing image retrieval mechanisms support only QBE type query interfaces [FSN+95] or SQL like query interfaces [SFV, SC97a, OS95], the drawbacks of both are listed. Actually, QBE type query interfaces should allow to browse the database. In the proposed prototype system, an initial query interface takes over some steps in the search path and accelerates the process (Figure 1.2). Navigational mode of query, i.e. browsing, enables query refinement mechanism, because the next query is based on results of the previous query. The browser starts with little specification and tries to explore images by reducing from a large set to a small set. Moreover relevance feedback is provided to the user, so that s/he can refine the queries according to matching image features. None of the image retrieval systems have such kind of feedback mechanism. Some of image retrieval systems just provide query results with some relevant features but not with a similarity percentage with respect to the query term.

6.2 Image Features

Images are a kind of multimedia data (refer Figure 1.1), whose encoding is defined by standard formats, such as Bitmap, GIF, JPEG and MPEG. The storage representation of images is basically a direct translation of the image, pixel by pixel. So there is no concept of a shape, or texture, unless it is in *vector* format. Since images cannot be described by basic components such as lines, finding such objects or more complex objects within an image is non-trivial.

Image data requires its content to be interpreted for querying. Therefore content-based queries require sophisticated indexing schemes and content-analysis algorithms to generate content descriptions. These content descriptions should cover the numerous features of images.

Proposed Solution

A high-level abstraction of visual information is called an image feature, and there are several image features, e.g. image format, bit depth, image size, compression, color, color format, texture type, texture dimension, texture color, texture's edge density, shape type, shape orientation, shape color, spatial, etc. These multi-dimensional features of visual information can be classified in four groups: functional, constant, variable, and predicative. Each feature is taken as a dimension. The main advantage of this classification is to reduce the number of feature dimensions and to omit from the beginning irrelevant features in queries. In some cases, it is spurious to use more features if they are derivable from existing features, e.g. for spatial features, not all combinations of relative object positions are needed. Some are computable in terms of other positions [Sam95].

Functional Features

Functional features are derivable from an image by some mathematical function or algorithm. For example, color distribution is represented through 3-D histograms in RGB color space. Extraction of shape information is based on thresholding. When the shape of the object is arbitrary, either its point of inertia or the minimum bounding box should be used to calculate its spatial location. For texture recognition, co-occurrence matrices are calculated using the periodic patterns in the image, and these matrices are used for similarity matching in N-dimensional space. *Color, Shape, Texture* and *Spatial* features are called *basic image features* and they are widely used in image retrieval systems.

Color feature: Color histogram, color space, color location.

Shape feature: Shape type (e.g. circular, freehand, fractal, amorphous, etc.), number of edges (e.g. straight line, triangle, polygonal, etc.), shape location (e.g. (x,y) coordinates of the bounding region), shape color (color attributes are covered), shape size, orientation.

Texture feature: Texture dimension (horizontal, vertical textures are 1-D; multiple oriented, stochastic textures are multi-D), texture color (all color attributes are covered), texture location, edge density, randomness in the pattern, size.

Spatial feature: Relative position to a reference object (e.g. above, below, left, right, overlaps, adjacent), absolute position (e.g. X-position, Y-position, height, width), orientation (e.g. North-south direction), point of inertia.

Constant Features

The values of constant attributes are assigned when the image is entered into the database and they do not change throughout the lifetime of the image unless the image is deleted or updated.

Attributes to view the image: type (bit-mapped, vector), format (JPEG, GIF, etc.), compression (LZW, Wavelet, JPEG), size and expansion rate, color metric (RGB, CMYK with several bit depths).

Attributes about the image: image ID, source (X-ray, digitized, etc.), copyright information, resolution.

Variable Features

Associated text that subjectively describes the image content. For example, category, subject, importance of the image.

Predicative Features

Presence or absence of predefined patterns to accelerate the frequently searched features [OS95]. For example, *exist(Sun)* = true or false.

6.3 Approach to the Multi-dimensional Indexing Problem

The categorization of image features may not help much with the indexing problem. Therefore, some multi-dimensional indexing structures [Gut84, LJF94, WJ96, BKH96] are proposed in the literature, which are discussed in the previous chapters.

The problem of multi-dimensional indexing methods are tried to be solved by filtering the search space using the image signatures. This approach is one of the pioneering ideas of this work and of the prototype image search engine. According to the literature survey, none of the existing systems in the Internet use signature filtering for all of the basic image features. Moreover, none of the image retrieval systems or commercial applications employ TV-tree [LJF94] structure as an indexing structure for multimedia data.

The irrelevant images can be eliminated during the filtering phase and only relevant ones are inserted to the TV-tree structure. A fewer number of qualifying images gives rise to a fewer number of overlapping regions in the tree, i.e. a fewer number of paths should be searched unnecessarily.

The tree structure becomes shallower than before because there are a fewer number of entries than the case without filtering. So despite some unnecessary multi-path searches, it still spends less time than the previous case.

Moreover, the number of dimensions can be reduced by using a filtering mechanism for some dimensions, which decreases possibility of overlapping regions.

Image signatures are used for the removal of irrelevant data, rather than finding data, i.e. for filtering. The signature method is especially useful when an image has more than one signature for the same feature, e.g. when the image is separated into segments, each of which is indexed differently. In the retrieval phase, matching image segments are retrieved. Image signatures save more space than other structures (Refer to section 4.1.2).

For each basic image feature, a signature is computed using the image indexing information. In the implementation of the prototype image retrieval system, in order to avoid "false drops", only one bit is set for an individual color feature, for example red (without any spatial information) is represented as "0000 0000 0001". Superimposition (OR'ing) is used for combination of features for the same image. However the randomness rule of bit locations in the signature is violated and the signatures contain exact information. So the image signatures are turned to be image descriptors, which are actually bit vectors. In the prototype system, the bit-length of signatures used for color, shape and texture features are 12, 12 and 5 respectively.

In the prototype system, 11 main colors are selected because they are highly selective colors [BCGM97]. The experiments done during the work showed that only 3% of the 16.6 million colors can not representable with these 11 colors. For each of the eleven chosen colors, one bit is reserved and another bit is reserved for other remaining colors. So 12 bits are used for color information. There are 8 main shape types, 4 of them contain sub-options, so 12 bits are reserved for shape information. The last 5 bits are used for five different texture types.

6.3.1 Evaluation of the prototype system

There is no accurate performance values for the prototype system, but it retrieves different number of images or their segments depending on the distance calculation metric. According to some sample query tests, signature filtering improves the utilization of TV-tree nodes by 20%, and the number of nodes used in the TV-tree are decreased (i.e. shallower tree) The experiments have also shown that there is no *false drops* in the retrieved images, because exact

information of image features are reflected to their bit vectors. The matching image features are correctly retrieved, i.e. feedback mechanism works correctly.

The images are collected randomly from the Internet. Traffic signs are also included in the database in order to define their geometric shapes easily during indexing. The inaccuracy in color processing stems from the errors during image creation (e.g. noisy colors during scanning).

6.4 Image Search by Signature Filtering–SIS

The prototype system, called SIS, contains only a subset of the proposed solutions. For example, among all the image features only color, shape, texture and spatial features are selected in forming image signatures (image descriptors) due to their wide applicability and perceptibility. Moreover, these feature queries are also empowered by a keyword query part. These features are processed for signature computation and indexing in SIS's two main parts: image processing and indexing, and image search. The sample algorithms of SIS are in the Appendix F. The algorithms for other features (shape and texture) are similar to the given color algorithms.

6.4.1 Implementation

The main idea of this thesis is to approach to the content-based retrieval problem of image databases from an IR view and to use the latest multi-dimensional indexing structure, the TV-tree [LJF94] structure.

The original codes of TV-tree insertion and TV-tree search are adapted with additional functions to be compatible with the implementation. For example, the similarity distance function and the node class are implemented. Similarity distance function contains color, shape and texture similarity section (Appendix F.7). For color similarity histogram comparison is applied, which will be explained in the following sections and in Appendix F.3. Spatial features are compared by *AND*'ing their spatial signatures. In case of a mismatch

the distance parameter is increased with a penalty (e.g. specified query term weight for that feature). Only the images whose distance values are less than the distance number are taken as the qualified ones and they are retrieved. The distance number can be modified for each level of accuracy.

In order to extract color information from the images, they should be pre-processed for transforming the pixel's color information to image signatures (image descriptors) and to feature vectors (Appendix F.2). The computed image descriptors are then used for filtering the search space.

Image feature vectors are also computed during the pre-processing phase. They are used as the real entries in the multi-dimensional TV-tree structure. The prototype image search engine, called SIS (**S**ignature based **I**mage Filtering and **S**earch), facilitates querying based on image content within a randomly selected sample image set.

The implementation of SIS is divided into two main parts; image indexing for new image entries and image retrieval by restricting image features. In both parts, a Common Gateway Interface (CGI) program is written, which is accessible from any Web browser on the World Wide Web. Sample algorithms are given in the Appendix F.

Common Gateway Interface

CGI is a standard for interfacing external applications with information servers, such as HTTP or Web servers. A plain HTML document that the Web deamon retrieves is static. A CGI program, on the other hand, is executed in real-time, so that it can output dynamic information.

SIS is implemented on the Solaris system and C is chosen as the CGI programming language, because of the following advantages:

- C code occupies minimal space compared to interpreted languages. Command line options help reduce the binary code further and create optimized code.

- Execution time of compiled code is faster than the code of interpreted languages.

Query Interface

The interface between the CGI program and the user is prepared by using the HyperText Markup Language (HTML). HTML is a simple markup system used to create hypertext documents that are portable from one platform to another. *HTML forms*, which allow users to enter data, are used as an interface to the implemented part of the CGI program. The *HTML form* obtains query terms and the CGI program performs the computations on that information. The query results are then written by the same CGI into another HTML document, which is shown on the screen.

A sample HTML form and its appearance on the screen is given in the Appendix A and B.

Sample screen shot of SIS is divided into parts which are provided at the end of this chapter in Figures 6.6, 6.7, 6.8, 6.9, 6.10, 6.11.

Data Exchange

In SIS, the *POST* method is used in order to allow information flow between users and the CGI program, because the size of information to be exchanged is high. The following HTML line defines the type of method;

<FORM METHOD="POST" ACTION="/cgi-bin/caglar1.cgi">

In the first step the CGI program has to decode the user input and to derive the query term. Then it can apply the filtering mechanism and send the query to the database deamon and to the program for further processing.

In general, each input element is assigned a variable name, such as $entry_i$ for example. The user input is known as the $value_i$ associated with the names.

When the data are sent to a server, they are encoded as strings of the form:

$$name_1 = value_1 \& name_2 = value_2 ....$$

where $name_i$ are the names, and $value_i$ are the values selected by the user. The = and & characters have special meanings in this encoding scheme, and indicate the separators between variables. A sample program parses the input and executes the CGI program with the parsed user input 6.1.

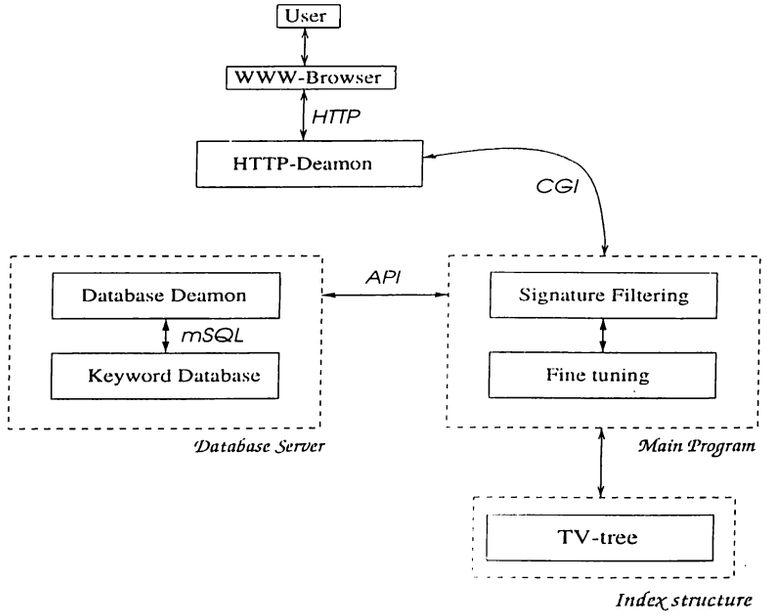

Figure 6.1: SIS Execution Strategy

Database Server Deamon

Mainly exact match queries are used in conventional database systems. Retrieval results are based on the Boolean value of *true* or *false*. There is no choice for *maybe*. In SIS, a keyword query screen is embedded because of the needs explained in the previous sections.

The entries in the keyword interface are parsed and each keyword is transmitted via the CGI program to the mSQL[1] database deamon to locate the

[1] mini-SQL (http://www.hughes.com.au)

"exact matches". The hits are returned to the CGI program by the mSQL database, where they are further transformed into a Web document to return the results of the search.

mSQL (mini-SQL) [Hug95] supports a subset of SQL as its query interface. There is no views, nested queries and functions (e.g. avg(), sum()). It provides data storage, manipulation and retrieval in table structures and joins between multiple tables. C language API of mSQL and and database engine work in a client/server environment over a TCP/IP network.

The mSQL deamon is a stand-alone application that listens for connections on a socket. It accepts multiple connections and serializes them. The keyword database is opened once, and kept open to avoid the time needed for every request. After the deamon has opened the database, it provides a socket where requests are sent. The output of the query can be returned to the CGI program via the socket.

Sample codes of insertion, deletion and table definitions are given in the Appendix C.

6.4.2 Image Processing and Indexing

In SIS, prior to insertion of an image into the database, while image features related to color attributes are extracted automatically, the indexer manually assigns texture and shape attributes of images through an easy-to-use graphical user interface. The GUI of indexing part is almost the same as the query screen, except the color part, because image color is indexed automatically.

A user can modify the index pattern of an already indexed image in order to refine the indexing scheme or to avoid unnecessary, detailed information in image features by using the indexing screen. This is because the results of current automatic shape or texture recognition are not very promising unless images have high contrast or noticeable color changes in shape boundaries, or in patterns.

A user should have an authorization to index an image in the database.

If s/he wants to access to the homepage of Image Index Screen[2], a password and an username is asked. After the validation of entries, s/he can select a specific region or the whole image to assign some index values. The color of the specified image segment or the whole image is automatically processed till the image indexing screen is ready on the screen.

Image color descriptors and indexes are created for each specified image section and the whole image. Meanwhile, a color histogram is computed for each image section.

Color Process

Color information is automatically processed for indexing and for image signature calculation, because images are actually composed of each pixel's color information. SIS accommodates three types of image format, the *PBM* (Portable BitMap), the *PGM* (Portable GrayMap) and the *PPM* (Portable PixelMap) for automatic image processing. All three image formats are just distinguished by the number and representations of colors used. *PBM*, *PGM* and *PPM* employ 2, 2 and 256 colors, respectively [WMB94].

These formats are widely used on Unix computer systems as the lowest common denominator in image format, and there are utilities, e.g. X-View, for converting between them and other common image formats.

PBM, *PGM* and *PPM* begin with an indicator that defines the number of colors, then give the width and height of the image in number of pixels. In *PPM*, the content of the pixelmap is followed by the number of colors available. In the case of *PGM* format, the maximum grayscale value, and finally the content of the graymap is given, where the maximum value, e.g. "255" means *black* and "0" means *white* . In *PBM*, the color contents are represented as a binary string of the number of pixel values. "1" denotes *black* and "0" denotes *white*. However, in *PPM*, color information of each pixel is expressed in *RGB* triplets. Table 6.1 shows an example of each format.

[2]http://www.nlp.cs.bilkent.edu.tr/~gunyakti/tez/Imagedatabase.html

PBM	P1 504 432 0 0 1 1 0 1 0 ...
PGM	P2 504 432 255 0 0 215 216 0 255 0 ...
PPM	P3 504 432 255 236_{Red} 230_{Green} 134_{Blue} 215_{Red} 208_{Green} 155_{Blue} ...

Table 6.1: PBM, PGM and PPM image data formats.

Although there are different color models, such as RGB (Red Green Blue), HSV (Hue Saturation Value) and CMYK (Cyan Magenta Yellow Black), the RGB model is chosen for SIS, since most of the image formats are based on this model.

Although the RGB model is 3-Dimensional, a function converts the 3-D space to the one dimensional vector v_{color}, which contains easily perceptible colors:

$$v_{color} = (red, green, blue, black, purple, yellow, pink, white, orange, gray, brown)$$

In the conversion function, some color ranges exploits the HSV values. Therefore a conversion function from RGB to HSV is also implemented using the algorithm in [Fe82].

Various researchers have proven that the human eye is not as sensitive to colors as to brightness. The experiments done during the work indicate that colors of an image discernible by the human eye account for about 97% of the total color content, because of the noisy data in images. Therefore the signatures and indexes are not created using all colors in the image, because of inaccuracy of human eye perception and incapability in digitizing the images.

In order to index images by color and to create color signatures, the RGB value of each pixel is converted by a function to the predefined eleven colors (yellow, orange, red, green, blue, purple, pink, brown, white, gray, black) and

one extra color for those that are not representable by the eleven colors. These eleven colors are selected, because they match human perceptual categories and help to distinguish objects from their environment [CO96], and from each other. Another bit reserved for color is used for colors that are either not representable with the existing 11 colors or less perceivable by the human eye, for example khaki and human's skin color. The color tests showed that only 3% of the 16.7 million colors cannot be represented by the selected eleven main colors, because of the gaps in the Table .

The table 6.2 denotes the RGB or HSV values of each color information. The values in table 6.2 are the extension of the values used in [BCGM97].

Color	RGB range	HSV range
Red	$(R > G + B) \wedge (R > G \simeq B)$	
Green	$(G > B) \wedge (G > R)$	
Blue	$B > \frac{R+G}{2}$	
Black	$R \simeq G \simeq B < 50$	
Purple		$[(258 < H < 310) \wedge$ $(((0.1 < S < 0.28) \wedge (0.88 < V)) \vee$ $((0.4 < S) \wedge (0.5 < V < 0.7)))]$ $\vee$ $[(320 < H < 330) \wedge$ $(((0.5 < S) \wedge (0.6 < V < 0.8)) \vee$ $((0.25 < S < 0.5) \wedge (0.41 < V < 0.78)))]$
Yellow	$(200 < R) \wedge (200 < G) \wedge (\mid R - G \mid < 15) \wedge$ $B < (0{,}8 * R)$	$(42 < H < 65) \wedge (0.3 < S) \wedge (0.89 < V)$
Pink		$(300 < H) \wedge (S < 0.87)$
White	$225 < (R \wedge G \wedge B)$	
Orange	$(240 < R) \wedge (130 < G < 170) \wedge (140 < B)$	$(5 < H < 43) \wedge (0.6 < S) \wedge (0.9 < V)$
Gray	$[\mid R - G \mid \wedge \mid R - B \mid \wedge \mid G - B \mid] < 20 \wedge$ $(R \vee G \vee B) < 200$	
Brown	$(100 \leq R \leq 180) \wedge (65 \leq G \leq 165) \wedge$ $(30 \leq B \leq 150)$	

Table 6.2: RGB and HSV ranges for 11 main colors

Color Histogram

A color histogram contains information on color distribution. It simply counts the number of pixels of each color. However, it lacks the spatial information of colors. So, some spatial knowledge is also embedded into the color histogram. Otherwise, images with the similar color compositions may appear different to humans [HCP95]. For example the histogram and their spatial values are transformed into floating points numbers. The decimal parts denotes color's spatial location and the floating point part denotes the color histogram value.

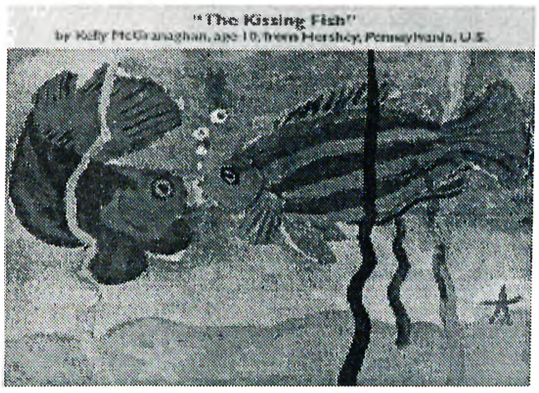

Figure 6.2: A sample image from the database: McGranaghan.jpeg

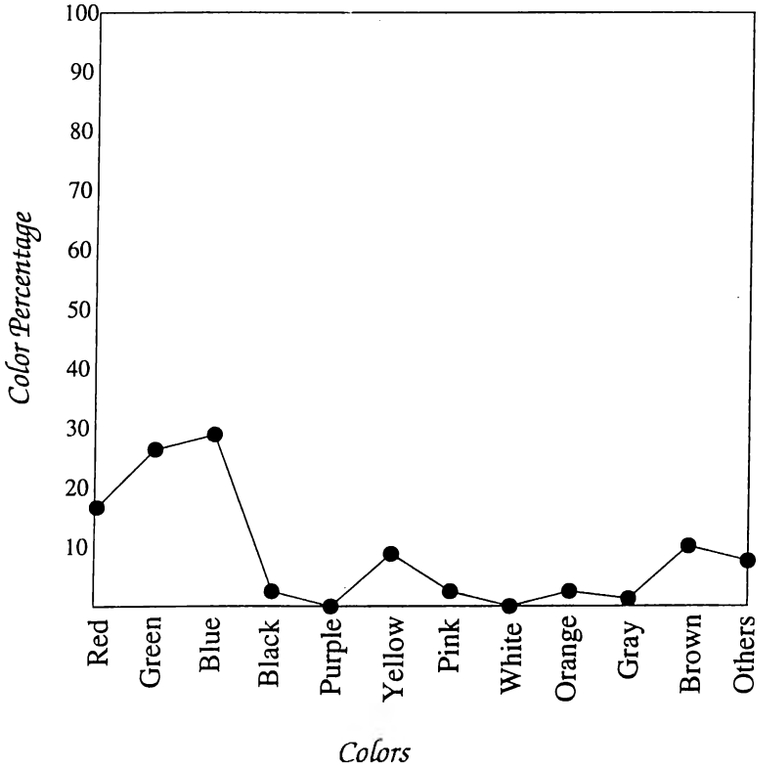

Figure 6.3: Color Histogram of McGranaghan.jpeg

The color histogram of $image_i$ is defined as a n-dimensional vector, where each dimension h_i represents the percentage of the $color_j$ used in the $image_i$.

$$Histog_{color}(i) = (h_1, ..., h_n) \; where \sum_{i=1}^{n} h_i = 1$$

A color histogram is used for fine-tuning the queries.

L_1-norm [SB91] is used to calculate the similarity distance, SD, between the color histograms, $Histog_i^{color}$ and $Histog_j^{color}$. I prefer L_1 to L_2 [FBF+94], since the former needs less computation.

$$SD_{L_1}(Histog_i^{color}, Histog_j^{color}) = \sum_{k=1}^{n} \mid i_k - j_k \mid$$

The L_1-distance between two histograms is always less than 2 and the L_2-distance is less than $\sqrt{2}$.

Texture & Shape Processing

Texture is actually the repeating pattern of an image. A texture may be found in different parts of an image or embedded in the background of the image objects. It is hard to extract texture patterns from images, because the notion of texture encompasses very irregular arrangements. The essentials of texture extraction algorithms are to detect an object, e.g. a particular shape, in the image, to find matching elements with its neighboring objects and to group the matching elements. The system [BPJ93] is able to deal with the case where there are only small number of repeated elements, e.g. two eyes in face images.

The main focus of this work is not on the automatic extraction algorithms for texture and shapes. Therefore texture and shape information of an image is processed manually by an indexer. Shapes are categorized in eight main groups according to edge types (linear, circular, triangular, rectangular, polygonal) and some other criteria (freehand, fractal, amorphous). In addition, shape colors, types (solid, contour, or both), weights and spatial locations can be defined using the interface.

Some shapes have sub-features, for example "circular" shapes can be further divided into arc, oval, and circle. All of these are built into SIS's GUI to index them better and calculate the signatures with higher selectivity. The interpretation of texture patterns is harder than that of other image features. Therefore sample texture images are provided for easing the pattern definition in the image. Again weight and spatial features selection are also supported for indexing and signature creation.

6.4.3 Discussion on manual indexing

It is obvious that manual indexing is a time-consuming and cumbersome task, especially with extremely large image databases. The indexer should also have additional information about the content of the image, which can not be extracted using basic image features (color, shape, texture, spatial). However, computational and human judgments of similarity must be generally correlated. Without this, the images that computer retrieves will not be those desired by the user.

There is always a tradeoff between time consumption and accurate or perfect content indexing. For example, a picture taken in a meeting between Arafat and Netanyahu can be processed by color, shape and texture extraction routines. Although powerful face recognition algorithms can capture the faces of Arafat and Netanyahu, none of the existing algorithms can extract the event "Signature Ceremony of Peace in the Middle East" without any semantic link to the concept or associated text along with the image [Sri95]. Therefore keyword indexing part is embedded for some content-based queries.

6.5 Image Searching and Retrieval

Querying images by a textual query language like SQL will limit the expressive power of visual queries. Not all the features of an image are structurable into predefined fields, some are complex, nested, variable-length or subjective to define in words. Therefore SIS provides a graphical query interface for visual

queries, which is very similar to the indexing screen.

Moreover, SIS's GUI supports queries like "*Find me pictures which contain these features*". It does not matter whether it is a trademark, fingerprint, photograph, or fabric. SIS's content-based relevance feedback at each step helps users to refine queries while browsing. SIS also extends these query types with a keyword search capability.

6.5.1 Type of image queries in SIS

The user interface is especially useful for narrowing the search space in the following QBE type query phases. The qualifying images are presented with their similarity percentages and matching features in ranked order. According to this relevance feedback, the user may select one or more of the resulting images to pose another QBE query.

Computing the matching degree of two images using correlation requires many multiplications and spends far more time and space than the simple bit comparison performed by the image signature method. Signature comparison can be used to identify a small number of candidate matches from the main image database and if necessary, a standard multidimensional index matching algorithm can then be used to produce an exact degree of match in this reduced set.

The prototype image retrieval system, SIS, supports the frequently used query types in visual data retrieval, such as nearest neighbors, sub-pattern matching and exact match (keyword) queries, which are explained in the following paragraphs.

Keyword Query

It is optional to select a general topic to limit the search space. However the keyword part should be filled for searching with one or more keywords, which are separated with a comma (,), a quotation mark ('), or a space.

The keyword query part is especially useful, when stated features are insufficient for expressing the query, for example "Give me a picture taken when Challenger exploded on 1/28/1986". The keywords, "Challenger", "explosion", and the date may be used in keyword searching. This part is actually converted to a SQL query, and a mSQL [mSQ95] database server is used.

Nearest Neighbor Queries

Nearest Neighbor queries are used to retrieve images which are within a certain distance of given query specification. It is actually a similarity query. The number of resulting images is adjustable using the interface (Figure 6.11).

Sub-pattern Matching

All of the queries mentioned above are applicable not only to the whole image, but also for its segments that are a priori indexed (Figure 6.4 and 6.5). At each query the image segments are also checked.

Spatial Feature Query

An object captured in the image occupies some space. The coordinates of the space is defined as spatial data. Spatial data denotes actually geometric shapes, varying from lines to polygons. Spatial data should be organized based on all the spatial keys in order to be retrieved faster, i.e. organization is based on the occupied space. This type of organization reveals the need for spatial indexing, in which the dimensionality of representative points are too high.

In SIS, images are divided into 9 (=3*3) main image segments. Each segment is indexed using the *indexing screen* of SIS. Although the query terms do not match with features of the whole image, they may match with the image segments.

A spatial feature query is supported by dividing the whole image into nine (3x3) uniform rectangular sections (Figures 6.4 and 6.5). The selected image

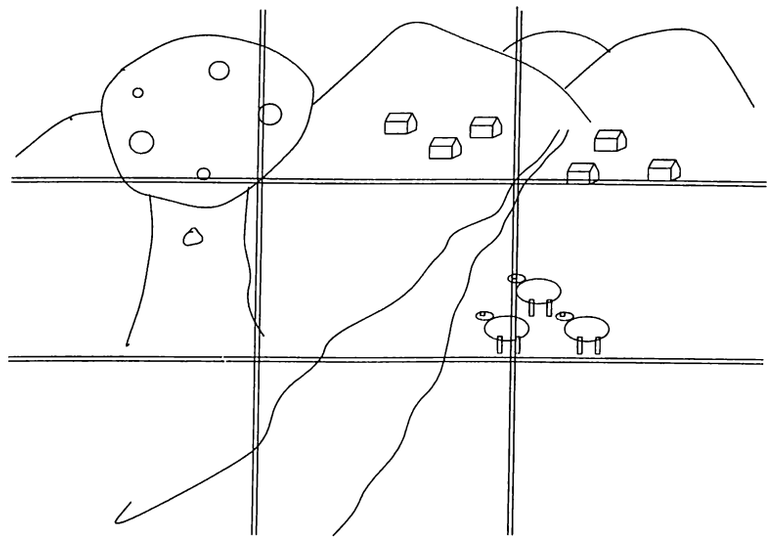

Figure 6.4: Image segments

1	2	3
4	5	6
7	8	9

Figure 6.5: Image segment numbers

sections are absolute locations. Each of 9 segments of the image, whole image or both can be indexed.

Images, whose signatures satisfy the query signature, are further searched in the multidimensional index structure for fine tuning and ranking. The search algorithm starts with the root of multidimensional index tree structure and examines each branch that intersects the search region, recursively following these branches.

6.5.2 Weighted Search

Various weights that are associated with image features are used in ranking, because content-based retrieval is not as precise as querying on record or object

attributes [KB96]. Query results are ranked from the most relevant one to the least relevant. The number of images to be retrieved can be set by the user. A similarity percentage is calculated as follows:

$$Similarity\_Percentage = \frac{\sum_i (weight(feature_i) * feature_i\_percentage)}{\sum_i weight(feature_i)}$$

$weight(feature_i)$: user defined weight of color, shape, texture and spatial features of images.

$feature_i - percentage$: calculated number while indexing images,e.g., color percentages in each image segment.

An example

We search images that have red, green and blue colors with weights of 1, 2, 3, respectively, i.e. the color percentage of blue is more than the other two colors (e.g. 10% red, 20% green and 30% blue). Images are processed by their color and it is realized that image A has 20% red, 30% green, 40% blue, and image B has 30% red, 40% green, and 20% blue. In this query $i = 3$, because we are querying only by three colors.

$$Similarity\_Percentage_A = \frac{(1 * 0.2) + (2 * 0.3) + (3 * 0.4)}{(1 + 2 + 3)} = 33.3\%$$

$$Similarity\_Percentage_B = \frac{(1 * 0.3) + (2 * 0.4) + (3 * 0.2)}{(1 + 2 + 3)} = 28.3\%$$

Although the sum of the percentages of red and green in image A is less than that in image B, it is denoted as more similar than image B, because of the weights that are defined while querying.

6.6 A sample image indexing and searching using SIS

To index the images only authorized users are allowed with a password mechanism explained in Appendix E. However, there is no need to be an authorized person to search images.

The shape, texture and keyword values of the sample image (Figure 6.2) are assigned manually using the Image Indexing Screen. This screen is created dynamically according to the selected image in the main database homepage[3] The color histogram of Figure 6.2 is calculated automatically by processing its .ppm format (Figure 6.3). The color signature (bit vector) is also calculated and reflected to the mSQL database.

The user can also assign weights to the shape or texture objects according to the subjective importance of the image feature. S/he can also rank the image features according to their importance by assigning them some weights.

To index Figure 6.2, oval shapes for fish, linear shapes for sea plants, polygonal shapes for the starfish can be selected and the keywords "fish", "starfish", "sea" can be selected.

If all the required index values are selected, the user can submit the index values by pressing the *Submit Index Values* button. At this stage the user input is processed:

- shape and texture signatures are calculated and stored temporarily in the mSQL database,
- other shape and texture information is interpreted and reflected to the mSQL database.

After submitting the index values for the selected image segment, the user is asked to *index another image* or to *start search*. According to the user's decision, a new page is created dynamically.

[3]http://www.nlp.cs.bilkent.edu.tr/~gunyakti/tez/Imagedatabase.html

If *start search* is selected the signature information of each image feature is reflected automatically to the files. An example of a color signature file is in Appendix D.

A user can search an image by specifying its basic image features or keywords using the interface. The weights in the interface are used for relevance feedback and to accelerate the search by discarding some noisy data.

The query terms are interpreted after firing the query, i.e. query signatures of basic image features are calculated. The query signatures are matched with the signature files. The image features of qualifying images are inserted to the TV-tree structure.

The values of satisfied basic image features (color, shape, texture) are encoded as floating point numbers and used as the real entries of the 29 dimensional feature vector for the representing image. 12 dimensions are reserved for color, 5 dimension for texture and 12 dimension for shape features. A solid, red arc (circular) with a weight of 3 in the second segment has $(10\mathbf{10}.100\mathbf{011})_2$ (i.e. $\langle spatial \rangle \langle \mathbf{type} \rangle . \langle color \rangle \langle \mathbf{weight} \rangle$) and $(10.11)_2$ (i.e. $\langle option \rangle . \langle weight \rangle$) as the feature vector entries for the shape type, circle.

The query feature vector for the given example in 6.5.2 is

$$\vec{q} = (511.1, 511.2, 511.3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

The first three dimensions contain the selected query term values for red, green and blue, respectively. Decimal part of each value in each dimension denotes the spatial signature (bit vector) of the feature and the floating part denotes the selected weight of the color feature. The rest are zero because they are not defined in the query term.

To calculate the distance between the query term and the entries in the TV-tree the steps in the Appendix F.7 are executed. Images that pass the step are returned to the user with their relevance information (Figure 6.13).

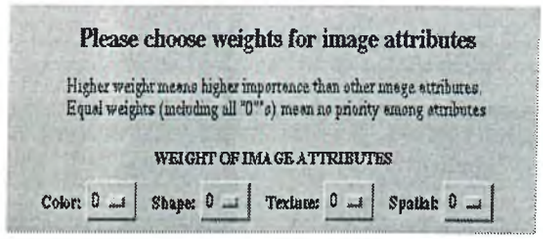

Figure 6.6: SIS' query weight specification

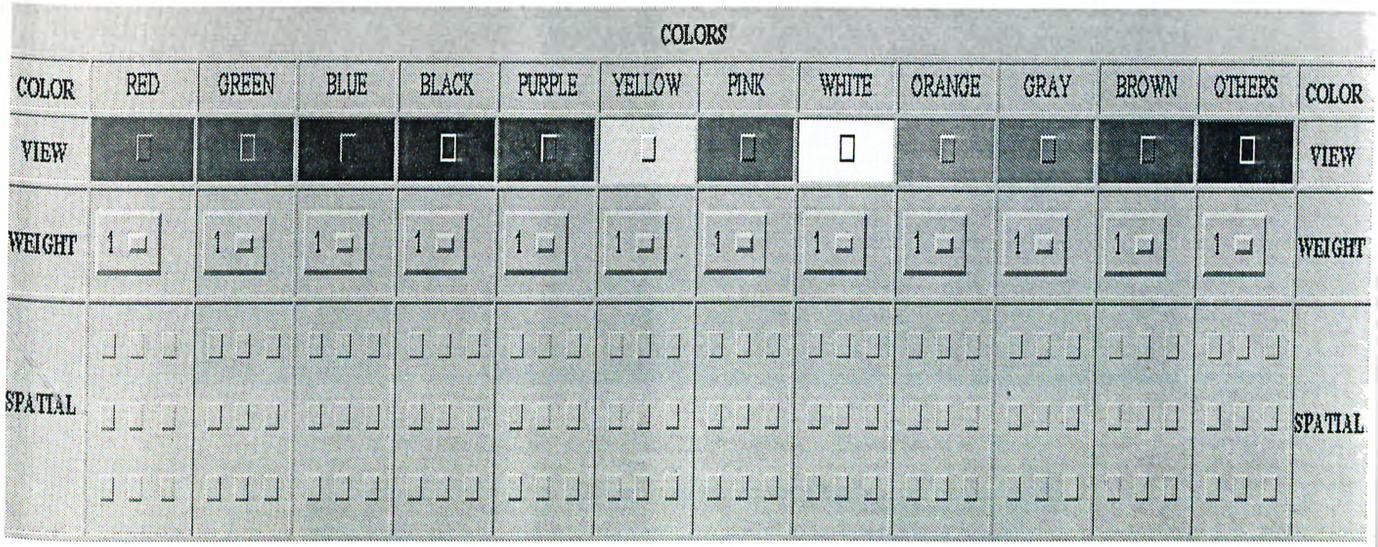

Figure 6.7: SIS Color Query Interface

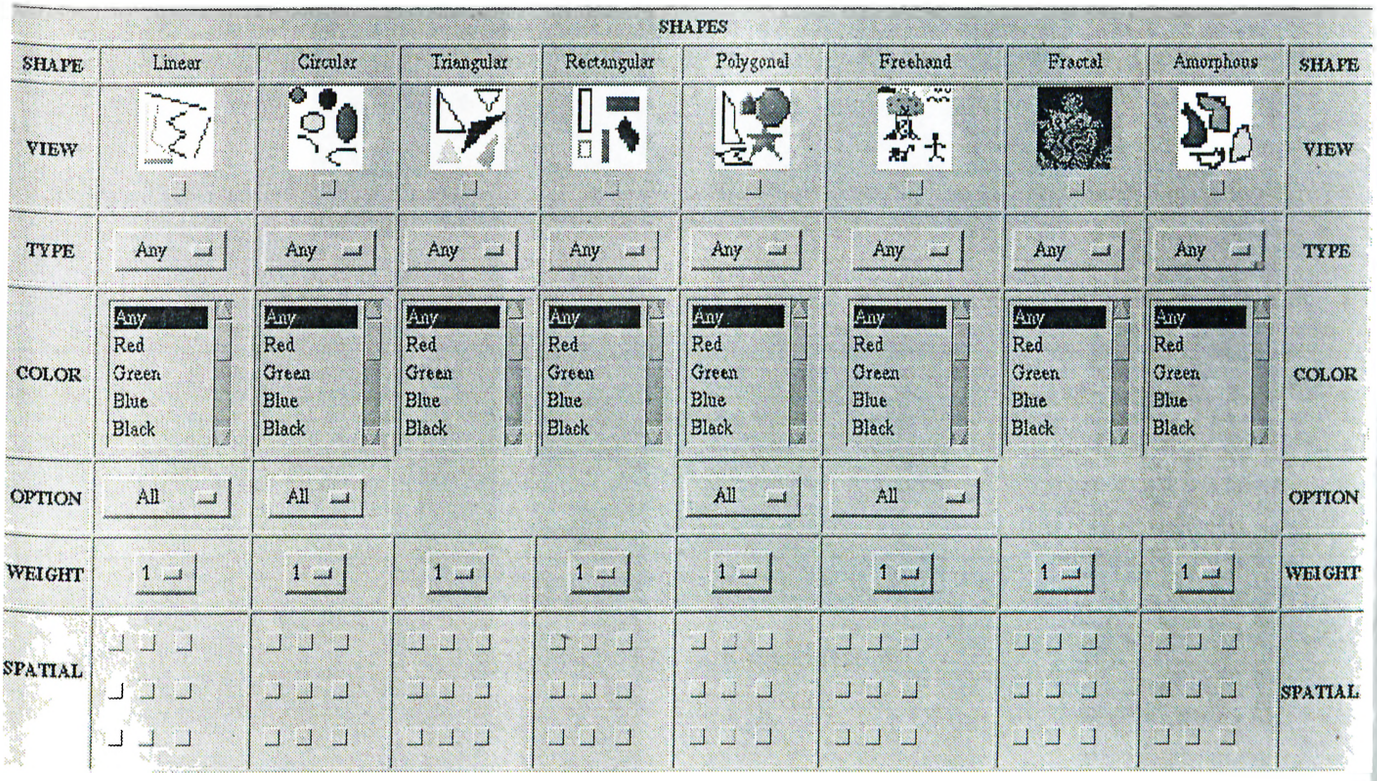

Figure 6.8: SIS Shape Query Interface

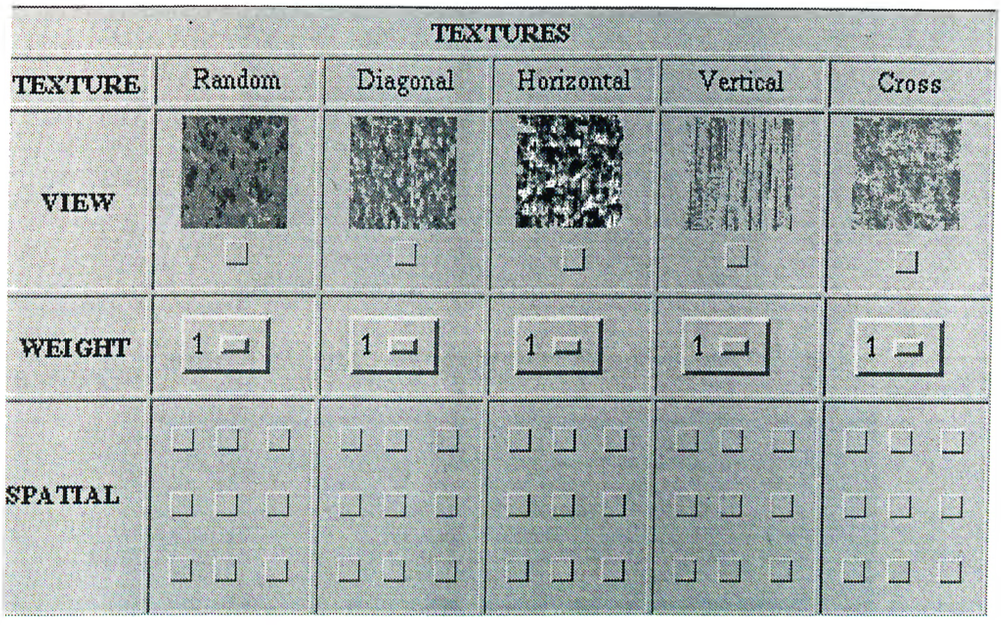

Figure 6.9: SIS Texture Query Interface

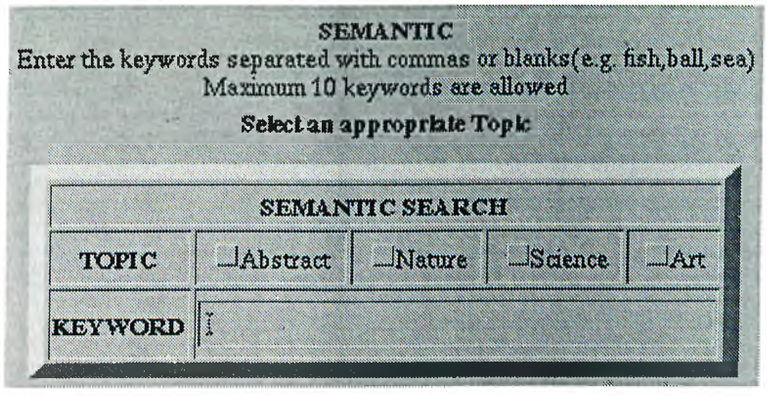

Figure 6.10: SIS Keyword Query Interface

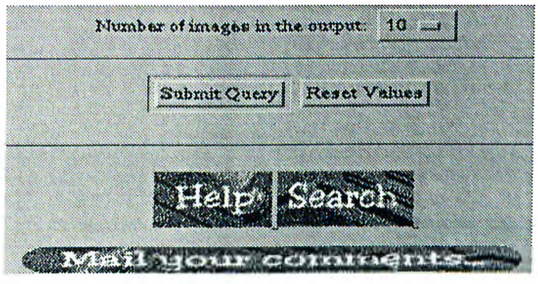

Figure 6.11: SIS Submit buttons

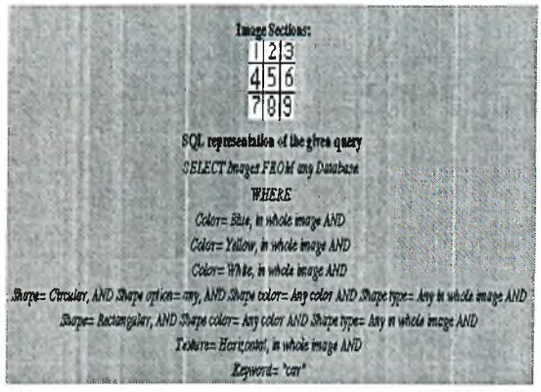

Figure 6.12: SQL like representation of query terms

Image	Image Name	Matching Percentage	Matching Partition [Upper-left][Lower-right]	Matching Feature
Size: 262x147 Select:	3x.jpg	42.86%	[175,49] [261,97]	COLOR Blue SHAPE Circular TEXTURE Horizontal KEYWORD car
Size: 262x147 Select:	3x.jpg	39.92%	[88,49] [174,97]	COLOR Blue SHAPE Rectangular TEXTURE Horizontal
Size: 239x215 Select:	Swans.jpeg	24.70%	[80,144] [159,214]	COLOR Blue SHAPE Circular
Size: 252x223 Select:	Haake.jpeg	23.96%	[84,75] [167,148]	COLOR Blue TEXTURE Horizontal
Size: 200x200 Select:	R3-1.gif	21.96%	[0,0] [65,66]	COLOR White SHAPE Circular

Figure 6.13: Output Screen & First Screen for QBE

Chapter 7

Conclusion and Future Work

The quality of stored descriptions, signatures, or features are determining factors of the power of image and multimedia databases. However, open questions are to decide what image features are valuable for indexing, how they should be extracted, compared and queried [Jai96]. The automatic indexing of image databases is also an unsolved problem. Moreover, developing powerful indexing and searching methods for multimedia data is still a progressing research area [BA95, Gha95].

Today, image labels, such as text identifiers, keywords, tags and other alphanumeric information provide valuable indexing keys, but do not fully represent the visual image content. Despite this problem, visual queries must be still integrated with text-based queries due to the guidance power of selective text.

The inefficiencies in multi-dimensional indexing are tried to be removed by employing image signatures. Although signatures are used in filtering color [FSN+95, FBF+94] and spatial orientations [LYC92] of image objects, they are not employed in shape and texture queries. Global image signatures are used for representations of the whole images, however in this thesis, as one of its distinguishing novelties, image segments are also used for indexing.

Our proposed system exploits traditional IR techniques and combines IR

with DBMS, where the most recent multi-dimensional indexing structure, TV-tree is applied. Easy to use GUI of SIS permits users to start an image search with visual image features (universal language). Then, SIS's content-based relevance feedback at each step helps users to accelerate the query refinements while browsing.

The main reason for using the HTML as an interface for the system is to take advantage of the fact that there exist Web browsers for each type of platforms. Therefore SIS serves its facilities to anyone who can access the Internet.

Regardless of all of its unsolved problems [Scl95], image search engines are vital tools to cope with the huge information flow while searching over the Internet.

7.1 Future Work

The following are some points for future directions of this work:

- In order to automatically process and index shape, texture and spatial features computer vision and pattern recognition algorithms can be applied. However, the perfect precision is beyond the capability of the existing algorithms.

- The user interface can be changed to a paint screen, where the user sketches the template of a query image rather than choosing the options. Due to the limitations of HTML this can not be implemented. Although the graphical functions of Java facilitates such kind of user interface, the implementation language should also be Java. So moving the implementation language from C to Java introduces a trade-off between a powerful user-interface and less performance in execution.

- SIS indexes only the images in the local database. The database will not depend on only local images, but also all the images on the Internet, if

a web traversing program is employed. A Web spider or WebCrawler is a tool, which actually traverses the Internet, captures documents, and downloads to the local site to process.

- To adjust perceptual similarity between a human being and computer, some machine learning algorithms can be applied. This process will also improve the precision in the image retrieval. After firing a sample query, the domain expert is asked to rank the retrieved images. The feedback of the domain expert is then compared with the computed ranking and used in the refinement of automatic similarity distance computation. This process will teach the machine how to improve the similarity formulation.

- Although bitmap image data contain just formatting and pixel color information, the data about the image that is related to the image contents may be stored as formatted data (i.e. textual data) in the header portion of the image data file. Therefore Natural Language Processing (NLP) techniques can be applied to process the image content.

- An adaptive signature calculation which lowers *false drop* probability can be applied for images of different scopes.

- Another way to improve the retrieval time performance is to parallelize the multi-dimensional indexing algorithm. As it is stated in the previous chapters, the drawback of this type of algorithms is the multiple branch search that deteriorates the performance. In some cases, some misleading branches are searched due to the overlapping partitions. A parallized index search algorithm will search the qualifying branches of the tree in parallel.

Appendix A

Forms Interface

The forms interface allows the user to formulate a query using the following HTML 3.0 elements.

Input Fields

- **Text Field:** Allows for text input. It is used for defining keywords of the query term.
  Example: <INPUT TYPE="TEXT" NAME=*keyword* SIZE=100>
- **Check Box:** Allows selection of some of the choices.
  Example: <INPUT TYPE="CHECKBOX" NAME=*red* VALUE="*ok*">*Red*
- **Radio Button:** Allows selection of only one of the choices.
  Example: <INPUT TYPE="RADIO" NAME=*Relative* VALUE="*ok*"> *Relative*
- **Image Pixel:** Submits to the CGI program the coordinates of the clicked point on the image. **Example:** <INPUT TYPE="IMAGE" NAME=*coordinates* SRC="*image1.gif*">
- **Hidden Field:** No user interaction is required. it is especially useful to transmit information between different and dynamic HTML pages. It is used to pass some information from the initial screen of SIS to the QBE-type query screen.
  Example: <INPUT TYPE="HIDDEN" NAME=*Numofimages* VALUE= "10">

Selection

- **Single Selection from options:** Allows a user to select one element from the given option list. The specific types of shapes can be selected from given options.

  Example: <SELECT NAME=*Shapesubtypes* >
  <OPTION> *Arc*
  <OPTION> *Circle*
  <OPTION> *Oval*
  </SELECT>

- **Multiple Selection from options:** Allows a user to select one or more elements from the given option list. In shape's color section of SIS, user is allowed to choose multiple colors for the same shape types.

  Example: <SELECT MULTIPLE NAME=*ShapeColors* >
  <OPTION> *Red*
  <OPTION> *Green*
  <OPTION> *Blue*
  </SELECT>

Submit

- **Submit button:** Submits the values that are defined by the user to the CGI program to be parsed. There are two methods for posting the user input; *POST* method and *GET* method.
  Example: <INPUT TYPE="SUBMIT" NAME=*Submit* VALUE="*SubmitQuery*">

- **Reset button:** Resets the values that are previously selected.
  Example: <INPUT TYPE="RESET" NAME=*Reset* VALUE="*ResetValues*">

Appendix B

Sample HTML form

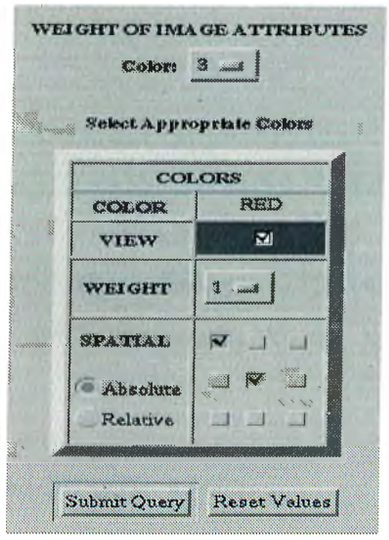

Figure B.1: HTML Form screen

This HTML form screen is generated using the following HTML code;

```
<HTML>
<HEAD>
   <TITLE>Image Search Engine by Caglar GUNYAKTI</TITLE>
</HEAD>
<BODY>
<CENTER>
```

```
<FORM METHOD="POST" ACTION="/cgi-bin/caglar1.cgi">

<B>WEIGHT OF IMAGE ATTRIBUTES</B> <BR>
<B>Color:</B>
<SELECT NAME="OptionCW">
<OPTION VALUE="0">0
<OPTION VALUE="1">1
<OPTION VALUE="2">2
<OPTION VALUE="3">3
<OPTION VALUE="4">4
<OPTION VALUE="5">5
</SELECT>
<P>

<TABLE BORDER=8>
<CAPTION ALIGN="top"><B>Select Appropriate Colors</B></CAPTION>
<TR><TH ALIGN="CENTER" COLSPAN=2><B>COLORS</B></TH></TR>
<TR ALIGN="CENTER"> <TD> <B>COLOR</B> </TD> <TD>RED</TD> </TR>
<TR ALIGN="CENTER"> <TD> <B>VIEW</B> </TD> <TD BGCOLOR=FF0000>
                    <INPUT TYPE="checkbox" NAME="Red," VALUE="ON">
</TD></TR>

<TR> <TD><B><CENTER>WEIGHT</B> </TD>
<TD>
<SELECT NAME="OptionRed">
<OPTION VALUE="1">1
<OPTION VALUE="2">2
<OPTION VALUE="3">3
<OPTION VALUE="4">4
<OPTION VALUE="5">5
</SELECT> </CENTER>
</TD>
</TR>

<TR> <TD><CENTER><B>SPATIAL </B></CENTER><BR>
```

```
<INPUT TYPE="radio" Name="Absolute" VALUE="0" CHECKED> Absolute
<BR><INPUT TYPE="radio" Name="Relative" VALUE="1"> Relative </TD>

<TD><TABLE>
<TR> <TD> <INPUT TYPE="checkbox" NAME="1" VALUE="0"></TD>
<TD><INPUT TYPE="checkbox" NAME="2" VALUE="0"></TD>
<TD><INPUT TYPE="checkbox" NAME="3" VALUE="0"> </TD></TR>
<TR><TD> <INPUT TYPE="checkbox" NAME="4" VALUE="0"></TD>
<TD><INPUT TYPE="checkbox" NAME="5" VALUE="0"> </TD>
<TD><INPUT TYPE="checkbox" NAME="6" VALUE="0"> </TD></TR>
<TR><TD> <INPUT TYPE="checkbox" NAME="7" VALUE="0"></TD>
<TD><INPUT TYPE="checkbox" NAME="8" VALUE="0"> </TD>
<TD><INPUT TYPE="checkbox" NAME="9" VALUE="0"> </TD></TR>
</TABLE> </TD> </TR> </TR></TABLE> <BR>

<INPUT TYPE="submit" NAME="Submit" VALUE="Submit Query">
<INPUT TYPE="reset" VALUE="Reset Values"><BR>
</FORM> </CENTER>
</HTML>
```

Appendix C

Sample codes of mSQL's C-API

C.1 Sample E-R Diagram for Keyword Part

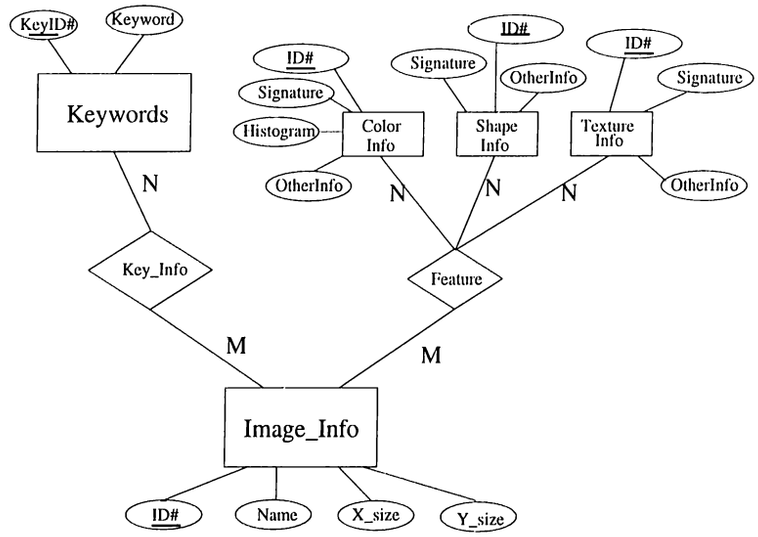

Figure C.1: E-R diagram of Keyword Query Part

Table definitions

```
create table Key_Info (
        imageid          int not null,
        keywordid        char(10) not null)

create table Keywords(
       keywordid        int not null,
       keyword          char(15) not null)

create table ImageInfo (
        imageid          int not null,
        imagename        char(20),
        title            char(20),
        artist           char(20),
        copyrightowner   char(20),
        dateentry        int,
        xsize            int,
        ysize            int,
        colorinfo        char(3)
)
```

Insert syntax

```
insert into Key_Info values ('12','aquarium')
```

Delete syntax

```
delete from  Key_Info where imageid > 12
```

A sample search program

```
void  Keyword_Query(char akeyword[])
/*
 This procedure prints all the imagenames, image ID's which have the
 same keyword, akeyword.
*/
{
#define SELECT_QUERY  "select Key_Info.keywordid, Key_Info.imageid,
                              ImageInfo.imagename
                       from   Key_Info, ImageInfo
                       where  Key_Info.keywordid = '%s' AND
                              ImageInfo.imageid =Key_Info.imageid"
int       count,i, sock,num;
char      qbuf[500];
m_result  *qresult;
m_row     qrow;
char      dbname[32]="image_tags";

/* Connects to the keyword database */
   if ((sock = msqlConnect(NULL)) < 0) {
     fprintf(stderr,"Couldn't connect to engine!\n%s\n\n", msqlErrMsg);
     exit(1);
   }

/* Selects related one out of many databases */
   if (msqlSelectDB(sock,dbname) < 0) {
     fprintf(stderr,"Couldn't select database %s!\n%s\n",dbname,msqlErrMsg);
   }

   sprintf(qbuf,SELECT_QUERY,akeyword);

/* Perform query */
   if(msqlQuery(sock,qbuf) < 0)
     printf("Query failed (%s)\n",msqlErrMsg);
```

```
    else {
      qresult =  msqlStoreResult();

/* Each time one qualifying row is selected for processing */
      qrow    =   msqlFetchRow(qresult);
      if (qrow==NULL)
        printf("No satisfying result for the given keyword!\n");
      else {
        while (qrow!=NULL) {
          printf("Image id: %d Image name: %s Annotated Keyword:%s"
                                        ,atoi(qrow[1]),qrow[2],qrow[0]);
          qrow=msqlFetchRow(qresult);
        }
        msqlFreeResult(qresult);
      }
    }
    msqlClose(sock);
}
```

Appendix D

Color Signature File Sample

The signature files for shape and texture have the similar structure.

```
0                /* Image ID#       */
0 0 3 2          /* Segment Coordinates (upper-left, lower-right) */
2                /* Color Signature: Green */
4 0 6 2          /* Segment Coordinates (upper-left, lower-right) */
130              /* Color Signature: Green, White */
0 0 9 7          /* Segment Coordinates (upper-left, lower-right) */
130              /* Color Signature: Green, White */

4                /* Image ID#  of  McGranaghan.jpeg     */
0 0 251 215      /* Segment Coordinates (upper-left, lower-right) */
3111             /* Color Signature: Red, Green, Blue, Yellow,    */
                 /*                  Brown, Others                */
```

Appendix E

Authentication for Indexing

Authentication is supported using the authentication mechanism of Apache Web Server. Following file, named *.htaccess*, is located on the same directory as Imagedatabase.html resides.

```
AuthUserFile /csgrad/gunyakti/thesis/users
AuthGroupFile ../groups
AuthType Basic
AuthName Image Indexing Screen
require valid-user
```

In case of an access to the directory, user is asked to enter the authorized name and password, which are located in a directory which should be inaccessible by everyone for security reasons. The content of the password and the authorized user names are as follows;

```
gunyakti:aaesEooTPuayQ
arkun:aaOEBU/xtCMiA
reda:aaqhOYolPBsQM
```

The authorized user name is separated with a colon from the encrypted password. For encryption, *encrypt()* function of C is used. Only valid user name, password tuples are authorized.

Appendix F

Algorithms of SIS

ColorHistogram($imid$,sid): /* $imid$: Image ID number to be indexed */
/* sid: Segment ID number ($1 \leq sid \leq 9$)*/
begin
for each color c
$color[c] = 0$
Find_Segm_Coordinates($imid, sid, x_{upper\_left}, y_{upper\_left}, x_{lower\_right}, y_{lower\_right}$)
for each pixel p in ($x_{upper\_left}, y_{upper\_left}$),($x_{lower\_right}, y_{lower\_right}$)
c = Decide_color($R_{(x,y)}$,$G_{(x,y)}$,$B_{(x,y)}$);
$color[c] = color[c] + 1$
for each color c
$Histog_{(imid,sid)}[c] = \frac{color[c]}{size_{(imid,sid)}}$
return $Histog_{(imid,sid)}$
end.

Figure F.1: Color Histogram Computation Algorithm

```
ColorSignature(Histog_(imid,sid)):
/* Histog_(imid,sid) : Color Histogram of (imid, sid) pair */
begin
    sort Histog_(imid,sid)[i] in descending order
    sum = 0
    i = 0
    while (sum < 0.97) and (i < NUMBEROFCOLORS) do
        ColorSignature_(imid,sid)| = (1 << i)
        i = i + 1
        sum = sum + Histog_(imid,sid)[i]
    return ColorSignature_(imid,sid)
end.
```

Figure F.2: Color Signature Computation Algorithm

```
ColorDistance(Histog_(imid,sid),Histog_query_term):
/* Histog_(imid,sid) : Color Histogram of (imid, sid) pair */
/* Histog_query_term : Color Histogram of the query_term */
begin
    for each matching segment
        for each color c
            dist = abs(Histog_(imid,sid)[c] - Histog_query_term[c])
    return dist
end.
```

Figure F.3: Color Distance Computation Algorithm

```
IndexColor(imid, sid):
/* imid : Image ID number to be indexed */
/* sid: Segment ID number (1 ≤ sid ≤ 9)*/
begin
    histog = ColorHistogram(imid, sid)
    sign = ColorSignature(histog)
    Insert_to_DB(imid, sid,histog)
    Write_to_signaturefile(Signfile, sign, imid, sid)
end.
```

Figure F.4: Image color indexing algorithm of SIS

```
Indexing_Algorithm(imid, sid):
/* imid : Image ID number to be indexed */
/* sid: Segment ID number (1 ≤ sid ≤ 9)*/
begin
    Get_user_input
    if color_weight > 0
      IndexColor(imid,sid)
    if shape_weight > 0
      IndexShape(imid,sid)
    if texture_weight > 0
      IndexTexture(imid,sid)
    if number_of_keywords > 0
      InsertKeyword(keywords)
end.
```

Figure F.5: Image indexing algorithm of SIS

```
ShapeDistance(Img^i_shape, QT_shape):
/* Img^i_shape : The shape info. of image i */
/* QT_shape : The shape information of the query term */
begin
    shape_dist=0
    for each mismatching shape spatial location of Img^i_shape
        shape_dist += QT_shape.weight
    for each matching shape spatial location of Img^i_shape
        for each mismatching shape subattribute of Img^i_shape
            shape_dist += QT_shape.weight
    return shape_dist
end.
```

Figure F.6: Shape similarity distance calculation algorithm of for TV-tree

```
Distance_Calculation(Image_i, QueryTerm):
/* Image_i : A member of filtered image set */
/* QueryTerm : The query term */
begin
    distance = 0
    distance = ColorDistance(Histog_Image_i, Histog_QueryTerm)
               + ShapeDistance(Image_i.Shape, QueryTerm.Shape)
               + TextureDistance(Image_i.Texture, QueryTerm.Texture)
    return distance
end.
```

Figure F.7: Similarity distance calculation algorithm for TV-tree

```
Fine_Tuning(Result,QueryTerm):
/* Result : The set of filtered images */
/* QueryTerm : The query term */
begin
    Insert_TV_tree(Result)
/* distance : similarity distance number */
    Result = Search_TV_tree(QueryTerm,distance)
    return Result
end.
```

Figure F.8: Fine_tuning algorithm of SIS

```
Searching_Algorithm():
begin
    QueryTerm = Get_query_term
    Q_Sign= CalculateSignature(QueryTerm)
    Result = ∅
/* Result contains the filtered image set*/
    if query_color_weight > 0
      for each (imid, sid) pair in the database
          Result = Result ⋃ FilterColor(imid, sid,Q_Sign.color)
    if query_shape_weight > 0
      for each (imid, sid) pair in the database
          Result = Result ⋃ FilterShape(imid, sid,Q_Sign.shape)
    if query_texture_weight > 0
      for each (imid, sid) pair in the database
          Result = Result ⋃ FilterTexture(imid, sid,Q_Sign.texture)
    if query_number_of_keywords > 0
      for each (imid, sid) pair in the keyword database
          Result = Result ⋃ QueryKeyword(keywords,QueryTerm.keyword)
    Result = Fine_Tuning(Result,QueryTerm)
    Output(Result)
end.
```

Figure F.9: Image search algorithm of SIS

Bibliography

[AC93] D. Aktug and F. Can. Signature files: An integrated access method for formatted and unformatted databases. Technical report, Department of Systems Analysis, Miami University, May 1993.

[AK92] K. Aberer and W. Klas. The impact of multimedia data on database management systems. Technical Report ICSI-TR-92-065, International Computer Science Institute, Berkeley, CA, USA, 1992.

[BA95] G. Baxter and D. Anderson. Image indexing and retrieval: some problems and proposed solutions. *New Library World*, 96(1123):4–13, 1995.

[BCGM97] S. Belongie, C. Carson, H. Greenspan, and J. Mali. Recognition of images in large databases using a learning framework. Technical Report TR 97-939, UC Berkeley CS, 1997.

[BKH96] S. Berchtold, D. A. Keim, and H.P.Kriegel. The x-tree: An index structure for high-dimensional data. In *Proc. of 22th Int'l Conf on VLDB*, pages 28–39, Bombay, India, September 1996.

[BKSS90] N. Beckmann, H. P. Keiegel, R. Schneider, and B. Seeger. R* tree: An efficient and robust access method for points and rectangles. In *Proc. ACM SIGMOD Int'l Conf. on the Management of Data*, pages 322–331, 1990.

[BM72] R. Bayer and E. McCreight. Organization and maintenance of larger ordered indexes. *Acta Informatica*, 1(3):173–189, 1972.

[BPJ93] J. R. Bach, S. Paul, and R. Jain. A visual information management system for the interactive retrieval of faces. *IEEE Transactions on Knowledge & Data Engineering*, 5(4):619–628, August 1993.

[Cat96] Cath. From 'virtual librarian' to 'virtual curator'. In *Electronic Imaging & Visual Arts on EVA*, National Gallery, London, England, 1996. http://www.hart.bbk.ac.uk/ cath/eva96.htm.

[CD95] P. Constantopoulos and M. Doerr. An approach to indexing annotated images. In *Proc. Int. Conf. on Hypermedia & Interactivity in Museum-ICHIM '95*, San Diego, CA, USA, October 1995.

[Che86] Christodoulakis and et.al. Multimedia document presentation, information extraction and document formation in minos: A model and a system. *ACM TOOIS*, 4(4), 1986.

[CF95] K. I. Lin C. Faloutsos. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, volume 24, pages 163–174, San Jose, CA, USA, June 1995.

[CO96] C. Carson and V. E. Ogle. Storage and retrieval of feature data for a very large online image collection. *IEEE Data Engineering*, 19(4):19–27, December 1996.

[Com79] D. Comer. The ubiquitous b-tree. *ACM Computing Surveys*, 11(2), June 1979.

[CSY87] S. K. Chang, Q. Y. Shi, and C. W. Yan. Iconic indexing by 2d string. *IEEE Transactions on Pattern Recognition and Machine Intelligence PAMI*, 9:413–428, 1987.

[FBF+94] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems (JIIS)*, 3(1):231–262, February 1994.

[FC84] C. Faloutsos and S. Christodoulakis. Signature files: An access method for documents and its analytical performance evaluation. *ACM Transactions on Office Information Systems (TOIS)*, 2(4):267–288, October 1984.

[Fe82] Foley and Van Dam et.al. *Introduction to Computer Graphics.* Addison Wesley, 1982.

[FSN+95] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, P. Yanker, and D. Steele. Query by image and video content: The qbic system. *IEEE Computer*, 28(9):23–31, September 1995. Demo Version; http://wwwqbic.almaden.ibm.com/cgi-bin/QbicStable.

[Gar82] I. Gargantini. An effective way to represent quadtrees. *Communications of the ACM*, 25(12):905–910, 1982.

[GBD+94] A. Ghafoor, S. Baqai, S. Dagtas, Y. F. Day, M. Iino, and J. Yang. Distributed multimedia database management. Annual Research Summary, Purdue University, June 1994.

[Gha95] A. Ghafoor. Multimedia database management systems. *ACM Computing Surveys*, 27(4), December 1995.

[G.R92] G.R.Thoma. Image databases in medicine. NSF VIMS Worksho, March 1992.

[Gro93] The POSTGRES Group. *The POSTGRES Reference Manual.* Computer Science Division, University of California, Berkeley, Berkeley, CA, USA, January 1993.

[Gup95] A. Gupta. Visual information retrieval technology: A virage perspective. Technical Report TR95-01, Virage, Inc., 1995.

[Gut84] A. Guttman. R trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Int'l Conf. on the Management of Data*, pages 47–57, 1984.

[Haa93] K. Haase. Framer: A portable persistent representation library. *Proc. of the AAAI Workshop on AI in Systems and Support*, 1993.

[HCP95] W. Hsu, T.S. Chua, and H.K. Pung. An integrated color-spatial approach to content-based image retrieval. In *Multimedia '95*, pages 305–313, San Francisco, CA USA, November 1995. ACM.

[Hug95] D. J. Hughes. Mini sql: A lightweight database engine. In *QAUUG Summer Technical Conference*. QAUUG, April 1995. http://www.Hughes.com.au/library/msql1/qauug95.htm.

[Jai96] R. Jain. Infoscopes: Multimedia information systems. In B. Furth, editor, *Multimedia Systems & Techniques*, pages 217–254, MA, USA, 1996. Kluwer Academic.

[KB96] S. Khoshafian and A. B. Baker. *Multimedia and Imaging Databases*, chapter Querying and Content Retrieval in Multimedia Databases, pages 325–376. Morgan Kaufman Inc., 1996.

[KCH95] P. M. Kelly, M. Cannon, and D. R. Hush. Query by image example: the candid approach. In *Storage & Retrieval for Image & Video Databases III:SPIE*, volume 2420, pages 238–248, 1995.

[Kim] Y. M. Kim. Multimedia database systems. http://www.cs.rpi.edu/ kimy/multimedia.html.

[LJF94] K. Lin, H. V. Jagadish, and C. Faloutsos. The tv-tree - an index structure for high-dimensional data. *VLDB Journal*, 3(4):517–542, 1994.

[LOS95] J. Z. Li, T. Ozsu, and D. Szafron. Query languages in multimedia database systems. Technical Report TR 95–25, Department of Computing Science, University of Alberta, December 1995.

[LYC92] S. Y. Lee, M. C. Yang, and J. W. Chen. Signature file as a spatial filter for iconic image database. *Journal of Visual Languages and Computing*, 3:373–397, 1992.

[mSQ95] msql (mini sql) database, version 1.0.14. http://www.hughes.com.au, 1995.

[NHS84] J. Nievergelt, H. Hinterberger, and K.C. Sevcik. The grid file: An adaptable symetric multikey file structure. *ACM Transactions on Database Systems*, 9:38–71, 1984.

[NOL95] Y. Niu, M. T. Ozsu, and X. Li. A study of image indexing techniques for multimedia database systems. Technical Report TR 95–19, Department of Computing Science, University of Alberta, July 1995.

[OS95] V. E. Ogle and M. Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE Computer*, 28(9):40–48, September 1995.

[OSEMV95] M. T. Ozsu, D. Szafron, G. El-Medani, and C. Vittal. An object-oriented multimedia database system for a news-on-demand application. *Multimedia Systems*, 3:182–203, 1995. .

[PPS94] A. Pentland, R.W. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. In *Proc. Storage and Retrieval for Image and Video Databases II*, volume 2185 of *SPIE Conference Proceedings*, San Jose, CA, USA, February 1994.

[PS96] P.Pazandak and J. Srivastava. Requirements for mmdbms: A report. Distributed Multimedia Center, University of Minnesota, March 1996.

[RNL95] T. C. Rakow, E. J. Neuhold, and Michael Lohr. Multimedia database systems - the notions and the issues. In Georg Lausen, editor, *Datenbanksysteme in Buro, Technik und Wissenschaft (BTW)*, pages 1–29, Berlin, March 1995. Springer.

[Rob81] J.T. Robinson. The k-d-b-tree: A search structure for large multidimensional dynamic indexes. *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 10–18, 1981.

[Sam95] H. Samet. General research issues in multimedia database systems. *ACM Computing Surveys*, 27(4), December 1995.

[SB91] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[SC'96] J. R. Smith and S. F. Chang. Visualseek: a fully automated content-based image query system. In *Proc. ACM Int'l. Conf. Multimedia*, Boston, MA, November 1996.

[SC'97a] J. R. Smith and S. F. Chang. An image and video search engine for the world-wide web. In *IS&T/SPIE Proc., Storage & Retrieval for Image and Video Databases V*, San Jose, CA, USA, February 1997.

[SC'97b] J. R. Smith and S. F. Chang. Safe: A general framework for integrated spatial & feature image search. In *Electronic Proceedings : Workshop on Multimedia Signal Processing*, Princeton, New Jersey, USA, June 1997. IEEE Signal Processing Society.

[Scl95] S. Sclaroff. World wide web image search engines. Technical Report TR95-016, Boston University, CS Dept, Cambridge, USA, June 1995. Position Paper for NSF Workshop on Visual Information Management.

[SFV] M. J. Swain, C. Frankel, and V.Athitsos. Webseer: An image search engine for the world wide web. Submitted to Computer Vision and Pattern Recognition CVPR '97.

[SM] SQL-3/MM. http://www.statkart.no/isotc211/wg4/wg-n043.html.

[SRF87] T. Sellis, N. Roussopoulos, and C. Faloutsos. The r+ tree: A dynamic index for multidimensional objects. In *Proc. 13th Int'l Conf. on Very Large Database*, pages 507–518, 1987.

[Sri95] R. K. Srihari. Automatic indexing and content-based retrieval of captioned images. *IEEE Computer*, 28(9):49–56, September 1995.

[WJ96] D. A. White and R. Jain. Similarity indexing with the ss-tree. In *Proc. of the 1996 Int'l Conf. on Data Eng. (ICDE'96)*, pages 516–523. February 1996.

[WMB94] I. H. Witten, A. Moffat, and T. C. Bell. *Managing gigabytes: compressing and indexing documents and images*. Van Nostrand Reinhold, 1994.

[ZDS+] W. Zhang, S. Dickenson, S. Scarloff, I. Marsic, S. Hawkins, J. Feldman, and S. Dunn. Searching medical image databases by image content. http://www.caip.rutgers.edu/ marsic/qbic.html.