

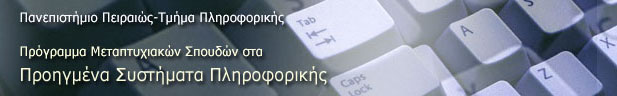
Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

|  |  |
| --- | --- |
| Τίτλος Διατριβής | Content Based Image Retrieval: A survey and an implementation |
| Ονοματεπώνυμο Φοιτητή | Xαραλάμπους Χαράλαμπος του Σταύρου |
| Αριθμός Μητρώου | ΜΠΣΠ-07026 |
| Κατεύθυνση | Συστήματα Υποστήριξης Αποφάσεων |
| Επιβλέπων | Τσιχριντζής Γεώργιος, Καθηγητής |



|  |  |
| --- | --- |
| Ημερομηνία Παράδοσης | Σεπτέμβριος 2009 |

|  |
| --- |
| Τριμελής Εξεταστική Επιτροπή |

|  |  |  |
| --- | --- | --- |
| (υπογραφή) | (υπογραφή) | (υπογραφή) |
| Όνομα Επώνυμο Βαθμίδα | Όνομα Επώνυμο Βαθμίδα | Όνομα Επώνυμο Βαθμίδα |

Abstract

With large image databases becoming a reality both in scientific and commercial domains, methods for organizing a database of images and for efficient retrieval have become important. We have worked on this problem providing a method for its solution that finds images approximately according to their similarity to the query image. Content Based Image Retrieval (CBIR from now on) is a technique for retrieving images on the basis of automatically-derived features such as color, texture and shape. CBIR differs from classical information retrieval in that image databases are essentially unstructured, since digitized images consist purely of arrays of pixel intensities with no inherent meaning. CBIR retrieves stored images from such a collection by comparing features extracted from the images themselves. Current indexing practices for images rely largely on text descriptors or classification codes, supported in some cases by text retrieval packages designed or adapted specially to handle images. Our system is a robust content-based image retrieval system based upon a combination of vision principles such as color, texture, and shape. Thus, using matching and comparison algorithms, the color, texture and shape features of one image are compared and matched to the corresponding features of another image. This comparison is performed using color, texture and shape distance metrics. In the end, these metrics are performed one after another, so as to retrieve database images that are similar to the query. The similarity between features was to be calculated using algorithms used by well known CBIR systems such as IBM's QBIC. For each specific feature there was a specific algorithm for extraction and another for matching.

# Introduction

CBIR operates on a totally different principle from keyword indexing. Primitive features characterizing image content, such as colour, texture and shape, are computed for both stored and query images, and used to identify (say) the 20 stored images most closely matching the query. Semantic features such as the type of object present in the image are harder to extract, though this remains an active research topic.

**Three commercial CBIR systems** are now available – IBM’s QBIC, Virage’s VIR Image Engine, and Excalibur’s Image RetrievalWare. In addition, demonstration versions of numerous experimental systems can be viewed on the Web, including MIT’s Photobook, Columbia University’s WebSEEk, Carnegie-Mellon University’s Informedia, Blobworld from University of California, Berkeley, SIMPLIcity [3] and ISTORAMA from Informatics & Telematics Institute in Thessaloniki. CBIR systems are beginning to find a foothold in the marketplace, prime application areas include crime prevention (fingerprint and face recognition), intellectual property (trademark registration), journalism and advertising (video asset management) and Web searching.

The **effectiveness** of all current CBIR systems is inherently limited by the fact that they can operate only at the primitive feature level. None of them can search effectively for, say, a photo of a dog – though some semantic queries can be handled by specifying them in terms of primitives. A beach scene, for example, can be retrieved by specifying large areas of blue at the top of the image, and yellow at the bottom. There is evidence that combining primitive image features with text keywords or hyperlinks can overcome some of these problems, though little is known about how such features can best be combined for retrieval.

**CBIR** draws many of its **methods** from the field of image processing and computer vision, and is regarded by some as a subset of that field. It differs from these fields principally through its emphasis on the retrieval of images with desired characteristics from a collection of significant size. Image processing covers a much wider field, including image enhancement, compression, transmission, and interpretation. While there are grey areas (such as object recognition by feature analysis) the distinction between mainstream image analysis and CBIR is usually fairly clear-cut. An example may make this clear. Many police forces now use automatic face recognition systems. Such systems may be used in one of two ways. Firstly, the image in front of the camera may be compared with a single individual’s database record to verify his or her identity. In this case, only two images are matched, a process few observers would call CBIR. Secondly, the entire database may be searched to find the most closely matching images. This is a genuine example of CBIR.

1. **Existing systems**

Here we provide a brief review of existing work on indexing and retrieval. Although the technology for organizing and searching images based on their content is still in its infancy, it shows a huge potential.

**IBM’s QBIC**

**IBM’s Almaden Research Center** has developed a system, **QBIC (Query By Image Content)**. It provides methods to query large on-line image database using image features as the basis for the queries. The image features they use include color, texture and shape of objects and regions. QBIC’s searches are approximate. These searches serve as “information filter” and they are interactive so the user can use visual query, visual evaluation and refinement in deciding what to discard and what to keep. The QBIC system has three functions: database population, feature calculation and image query. Since automatic outlining of image objects is not robust, object extraction in QBIC is done manually or semi-automatically (manually aided image analysis method). The feature calculation task is done by the system. The features include color, texture and shape. QBIC computes the color histogram as color features and its texture features are based on a three- dimensional feature space: coarseness, contrast and directionality. They choose this feature model mainly for its low computation complexity. For shape features, QBIC utilizes a combination of circularity, eccentricity, major axis orientation and a set of algebraic moment invariants. QBIC supports both “full scene” query (query by global similarity) and query by similar regions. The similarity functions it uses are mainly distance metrics, such as cityblock, Euclidean or weighted Euclidean distance. It computes the weighted histogram quadrature distance for color similarity. The texture distance is computed as weighted Euclidean distance in the three dimensional texture space. QBIC also supports “query by sketch” which works by matching the user drawn edges to automatically extract edges from images in database. Similarity query results correspond to nearest neighbor with the measures specified by the user in a certain user specified range or the result is an n-bestlist.

**MIT’s Photobook**

**MIT’s Media Lab** has developed an image and video clip query tool-**Photobook**. Like QBIC, it works by comparing features extracted from images. These features are parameter values of particular models fitted to each image. The models are color, texture and shape. The matching criteria include Euclidean, Mahalanobis, divergence, histogram, Fourier peaks and wavelet tree distances, or any linear combination of them. Photobook has three types of image descriptions with each of them handling a specific image content. Appearance descriptions (“Appearance Photobook”) are applied to face and keyframe databases, texture descriptions (“Texture Photobook”) are applied to Brodatz texture and keyframe databases, and shape descriptions (“Shape Photobook”) are applied to hand-tool and fish databases. The main feature of Photobook is that it includes FourEyes, an interactive learning agent which selects and combines models based on the example from the user. At the current stage, FourEyes is a tool for segmenting and annotating images. Though computer vision algorithms are not robust enough to automatically annotate general imagery, but they can be applied in certain cases or in some combinations. The problem is how to choose the model or algorithm. For example, in the case of vision texture, they defined a society of models including co-occurrence, random field, fractals, reaction-diffusion, eigenpatterns, morphology, Fourier bins, wavelets, steerable pyramids, autoregressive moving average, grammar, cluster-based probability, wold, particle systems, Gabor filters, etc. Since there is no one model that will be optimal, a semi-automatic tool, FourEyes, is incorporated to determine model and similarity measures appropriate for the task by learning from examples given by the user. When the user clicks on some regions and gives them a label, FourEyes can extrapolate the label to other regions in the image and in the database. It works by combining suitable models from its “society of models” according to the examples given by the user. FourEyes make Photobook a flexible environment which can support search using various features. But it does not offer assistance in choosing the right one for a given mission.

**Columbia’s VisualSEEK**

The **Center for Image Technology for New Media** in **Columbia University** has developed a image query system called **VisualSEEK**. VisualSEEK’s current version is mainly for searching images through the World Wide Web and it has an archive of 12,000 images. VisualSEEK consists of three parts:

1. The client application, which is a suite of Java applets that execute within the WWW browser. It collects the query from the user and generates the query string.

2. The network and communication application, which handles all communication across the WWW.

3. The server application, which receives the query string, executes the query and returns the results to the user. The server program generates Hypertext Makeup Language (HTML) code that displays the results of the query to the user. VisualSEEK currently supports global image similarity retrieval and retrieval by local or regional features (color and texture). Its search is different from QBIC and Photobook in that the user can query for images using both the visual properties of regions and their spatial layout. As for global image similarity retrieval, VisualSEEK has several similarity measures which can be chosen by the user, including: color histogram intersection, color histogram moments, color region intersection, color region Euclidean, color histogram Euclidean, texture set intersection, texture histogram intersection, texture and color intersection, texture and color Euclidean and color histogram quadratic(ordered by complexity of algorithm). For query by region features and spatial relationship, the system automatically extracts salient color and texture regions from an image and performs query on features and relative locations. The region extraction technique adopted by VisualSEEK is back-projection of binary color and texture sets.

**Berkeley’s Digital Library Project**

The **Computer Science Division** of the **University of California at Berkeley** is conducting research on digital libraries. Their goal is to provide a general framework for content-based image retrieval which allows search from low-level to high-level. They define a blob world representation, which is a transition from the pixel data to a small set of localized coherent regions in local color and texture space. At this level, it is quite similar to VisualSEEK and Photobook. The difference is that they try to categorize images into visual categories, which needs a learning process to give probabilistic interpretations of the blob regions in an image. The system also incorporates spatial information in its learning process. The color space they use in their processing is HSV because it is perceptually meaningful and can aid grouping and recognition. They incorporate a list of 13 colors and create a lookup table to divide any image into these color channels. Their texture feature extraction is based on information from “windowed image second moment matrix”, which can classify 6 kinds of texture types (non-texture, 2D texture, 4 1D texture in different directions). After obtaining color and texture features, they adopt an Expectation Maximization algorithm to get the segmentation of the image. The advantage of Expectation Maximization algorithm is that it can avoid fragmentation of main regions in segmentation. The learning process to generate categorization is based on a Bayes classifier, which uses blob’s color, texture and spatial location. Their experiment is done on about 1,200 images falling into 12 categories. The training and testing sets for Bayes classifier contain 2/3 and 1/3 of the images. The performance is quite consistent with visual perception although there are some misclassification among certain categories.

**Virage**

**Virage Inc.** has its roots in the research done at **University of California at San Diego**. Their technology of image retrieval is built around a core module called the Virage Engine. Virage Engine is an open, portable and extensible architecture to incorporate any domain specific schema. Virage applies three levels of information abstraction for images: the raw image(the image representation level), the processed image(the image object level) and the user’s feature of interest (the domain object level). The computed visual features are named by Virage as “primitives”, which are either global or local. According to Virage, a primitive should be meaningful for perception, compact in storage, efficient in computation, accurate in comparison and should be indexable. Several “universal primitives” given by Virage include global color, local color, structure and texture. These primitives are universal in the sense that they are useful in most domain- independent applications and they can be automatically computed by Virage Engine. The user can choose to mix and match these primitives in conjunction with domain specific primitives to build a particular application, such as medical or multimedia application. Its extensible architecture enables the user to “plug-in” domain optimized algorithms to solve specific problems. Virage Engine is delivered as a statistically or dynamically linkable library for various platforms (Sun, SGI, Windows and MAC). The library accommodates different types of databases and application frameworks. Databases such as Oracle, Sybase or image- manipulation and processing tool like Photoshop, CorelDraw could use the Virage Engine for image search and management. An Internet search engine like WebCrawler or InfoSeek can extend their capabilities with image finding in the network and thus help building a searchable image storage system distributed over the WWW.

**KRDL’s CORE**

**CORE, a COntent-based Retrieval Engine**, for multimedia databases, was developed by RWC Lab of KRDL. CORE provides functions in multimedia information systems for multimedia objects creation, analysis, storage and retrieval. Some salient techniques in CORE include: multiple feature extraction methods, multiple content- based retrieval methods, a novel content-based indexing on complex feature measures using self-organizing neural networks and a new technique for fuzzy retrieval of multimedia information. CORE has three main modules:

1. Multimedia data analysis module (Analysis),

2. Query module (Retrieval and Indexing),

3. Customization module (Training).

CORE provides elementary multimedia classes as building blocks for application system development. Its image representation scheme consists of four levels from bottom to top: gray/color image, segmented image, descriptions and features, interpretation. CORE provides three types of segmentation functions, namely, color segmentation, morphological egmentation, and segmentation by boundary extraction. Feature extraction functions are invoked to compute feature measures on segmented objects/ regions. These functions include: principal component analysis, moment invariant, shape measure by Fourier descriptors, projections, shape parameters, texture energy, color feature measures etc. CORE’s content-based indexing is located between feature and interpretation level. It utilizes the concept of tree classifier and content-based indexing, and uses LEP (Learning based on Experiences and Perspective) neural network model to provide a solid theoretical basis and to be able to fuse composite feature measures. This is used to generate self-organizing nodes for content-based index tree of complex and composite feature measures. This indexing scheme has been implemented in CORE to support similarity retrieval. It can also be used for fuzzy indexing on multi-variate fuzzy membership functions. There are four types of retrieval methods in CORE, namely, visual browsing, similarity retrieval, fuzzy retrieval and text retrieval. Similarity retrieval provides access to multimedia objects through feature measures, while fuzzy retrieval and text retrieval provide access through the interpretation of multimedia objects. User feedback is often necessary for effective retrieval. By using feedback functions, user can select one or more objects from current query results, which he/she thinks are very close to the desired one. Feedback function then finds the information from the selected objects and refines the query. Feedback function in CORE assumes that feedback objects are selected in a sequence such that the most desired one is selected first, and the selection will be based on the most similar features among those selected objects.

CORE has been used in the development of two application systems:

Computer- Aided Facial Image Inference and Retrieval (CAFIIR)

and

System for Trademark Archival and Registration (STAR)

In addition, a medical image database system for computer-aided diagnosis and surgery is under development using this engine. CORE has rich and comprehensive functionalities. However, each application has its domain specific problems so domain expertise must be added to customize the indexing and retrieval module. That is, training is a crucial step in the application development. We have developed our research using useful results and functions provided by CORE.

## Applications of Content-Based Retrieval

In [31], we see three broad categories of user aims when using a CBIR system. There is a broad class of methods and systems aimed at browsing through a large set of images from unspecified sources. Users of search by association **at the start** have no specific aim other than find interesting things. Search by association often implies iterative refinement of the search, the similarity or the examples with which the search was started. Systems in this category typically are highly interactive, where the specification may by sketch [30] or by example images. The oldest realistic example of such a system is probably [88]. The result of the search can be manipulated interactively by relevance feedback [68]. To support the quest for relevant results, other sources than images are also employed, see for example [168].

**Another class of users** aims the search at a specific image. The search may be for a precise copy of the image in mind, as in searching art catalogues, e.g. [48]. Target search may also be for another image of the same object of which the user has an image. This is target search by example. Target search may also be applied when the user has a specific image in mind and the target is interactively specified as similar to a group of given examples, for instance [31]. These systems are suited to search for stamps, art, industrial components, and catalogues, in general.

**The third class** of applications, category search, aims at retrieving an arbitrary image representative of a specific class. It may be the case that the user has an example and the search is for other elements of the same class. Categories may be derived from labels or emerge from the database [170]. In category search, the user may have available a group of images and the search is for additional images of the same class [29]. A typical application of category search is catalogues of varieties. In [74], [79], systems are designed for classifying trademarks. Systems in this category are usually interactive with a domain specific definition of similarity.

## Related Work in CBIR

CBIR for general-purpose image databases is a highly challenging problem because of the large size of the database, the difficulty of understanding images, both by people and computers, the difficulty of formulating a query, and the issue of evaluating results properly. A number of general-purpose image search engines have been developed. We tried to make our survey as complete as possible.

The common ground for CBIR systems is to extract a signature for every image based on its pixel values and to define a rule for comparing images. The signature serves as an image representation in the view of a CBIR system. The components of the signature are called features. One advantage of a signature over the original pixel values is the significant compression of image representation. Actually, the main task of designing a signature is to bridge the gap between image semantics and the pixel representation, that is, to create a better correlation with image semantics. Existing general-purpose CBIR systems roughly fall into three categories depending on the approach to extract signatures: histogram, color layout, and region-based search.

### Histogram Search

Histogram search algorithms [4], [18] characterize an image by its color distribution or histogram. Many distances have been used to define the similarity of two color histogram representations. Euclidean distance and its variations are the most commonly used [4]. Rubner et al. of Stanford University proposed the earth mover's distance (EMD) [18] using linear programming for matching histograms.

The drawback of a global histogram representation is that information about object location, shape, and texture [10] is discarded. Color histogram search is sensitive to intensity variation, color distortions and cropping.

### Color Layout Search

The color layout approach attempts to overcome the drawback of histogram search. In simple color layout indexing [4], images are partitioned into blocks and the average color of each block is stored. Thus, the color layout is essentially a low resolution representation of the original image. A relatively recent system, WBIIS [28], uses significant Daubechies' wavelet coefficients instead of averaging. By adjusting block sizes or the levels of wavelet transforms, the coarseness of a color layout representation can be tuned. The finest color layout using a single pixel block is the original pixel representation. Hence, we can view a color layout representation as an opposite extreme of a histogram. At proper resolutions, the color layout representation naturally retains shape, location, and texture information.

However, as with pixel representation, although information such as shape is preserved in the color layout representation, the retrieval system cannot perceive it directly. Color layout search is sensitive to shifting, cropping, scaling, and rotation because images are described by a set of local properties [28]. The approach taken by the recent WALRUS system [14] to reduce the shifting and scaling sensitivity for color layout search is to exhaustively reproduce many subimages based on an original image. The subimages are formed by sliding windows of various sizes and a color layout signature is computed for every subimage. The similarity between images is then determined by comparing the signatures of subimages. An obvious drawback of the system is the sharply increased computational complexity and increase of size of the search space due to exhaustive generation of subimages. Furthermore, texture and shape information is discarded in the signatures because every subimage is partitioned into four blocks and only average colors of the blocks are used as features. This system is also limited to intensity-level image representations.

### Region-Based Search

Region-based retrieval systems attempt to overcome the deficiencies of color layout search by representing images at the object-level. A region-based retrieval system applies image segmentation [20], [27] to decompose an image into regions, which correspond to objects if the decomposition is ideal. The object-level representation is intended to be close to the perception of the human visual system (HVS). However, image segmentation is nearly as difficult as image understanding because the images are 2D projections of 3D objects and computers are not trained in the 3D world the way human beings are. Since the retrieval system has identified what objects are in the image, it is easier for the system to recognize similar objects at different locations and with different orientations and sizes. Region-based retrieval systems include the NeTra system [11], the Blobworld system [2], and the query system with color region templates [22].

The NeTra and the Blobworld systems compare images based on individual regions. Although querying based on a limited number of regions is allowed, the query is performed by merging single-region query results. The motivation is to shift part of the comparison task to the users. To query an image, a user is provided with the segmented regions of the image and is required to select the regions to be matched and also attributes, e.g., color and texture, of the regions to be used for evaluating similarity. Such querying systems provide more control to the user. However, the user's semantic understanding of an image is at a higher level than the region representation. For objects without discerning attributes, such as special texture, it is not obvious for the user how to select a query from the large variety of choices. Thus, such a querying scheme may add burdens on users without significant reward. On the other hand, because of the great difficulty of achieving accurate segmentation, systems in [11], [2] often partition one object into several regions with none of them being representative for the object, especially for images without distinctive objects and scenes.

Not much attention has been paid to developing similarity measures that combine information from all of the regions. One effort in this direction is the querying system developed by Smith and Li [22]. Their system decomposes an image into regions with characterizations predefined in a finite pattern library. With every pattern labeled by a symbol, images are then represented by region strings. Region strings are converted to composite region template (CRT) descriptor matrices that provide the relative ordering of symbols. Similarity between images is measured by the closeness between the CRT descriptor matrices. This measure is sensitive to object shifting since a CRT matrix is determined solely by the ordering of symbols. The measure also lacks robustness to scaling and rotation. Because the definition of the CRT descriptor matrix relies on the pattern library, the system performance depends critically on the library. The performance degrades if region types in an image are not represented by patterns in the library. The system uses a CRT library with patterns described only by color. In particular, the patterns are obtained by quantizing color space. If texture and shape features are also used to distinguish patterns, the number of patterns in the library will increase dramatically, roughly exponentially in the number of features if patterns are obtained by uniformly quantizing features.

## Organizing an image collection

Whilst this survey is primarily focused on techniques for the storage and retrieval of electronic images, it is useful to reflect on the traditional practices of picture and other manual collections of images and videos. Image collections of various types are maintained by a wide range of organizations, of all sizes and in a variety of sectors.

The image database that we used in our project contains 1000 8-bit uncompressed bit maps **BMP**s that have been selected from the Corel collection. The following figure depicts a sample of images in the database:

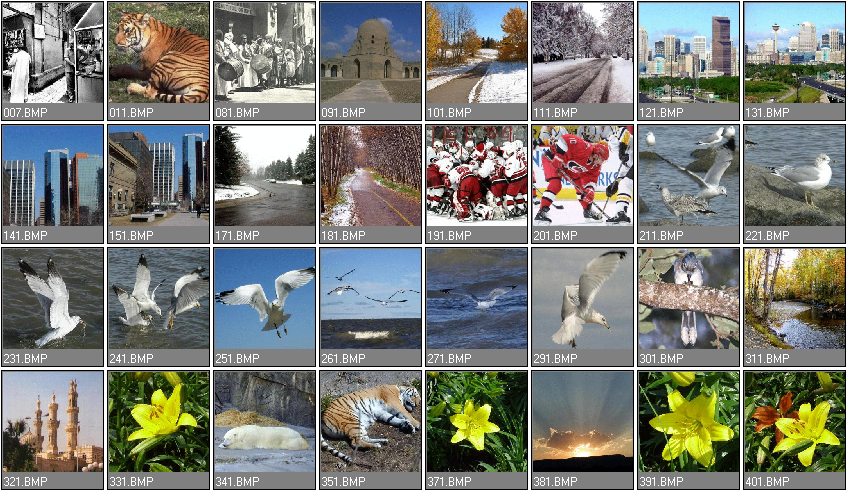


Figure: Image Database

### Proposed Solution

The solution initially proposed was to extract the primitive features of a query image and compare them to those of database images. The image features under consideration were color, texture and shape. Thus, using matching and comparison algorithms, the color, texture and shape features of one image are compared and matched to the corresponding features of another image. This comparison is performed using color, texture and shape distance metrics. In the end, these metrics are performed one after another, so as to retrieve database images that are similar to the query. The similarity between features was to be calculated using algorithms used by well known CBIR systems such as IBM's QBIC. For each specific feature there was a specific algorithm for extraction and another for matching.

# Color Space

Each pixel of the image can be represented as a point in a 3D color space. Commonly used color space for image retrieval include *RGB*, *Munsell*, *CIE L\*a\*b\**, *CIE L\*u\*v\**,*HSV* (or *HSL*, *HSB*), and opponent colorspace. There is no agreement on which is the best. However, one of the desirable characteristics of an appropriate color space for image retrieval is its uniformity[65]. **Uniformity** means that two color pairs that are equal in similarity distance in a color space are perceived as equal by viewers**. In other words, the measured proximity among the colors must be directly related to the psychological similarity among them**. RGB space is a widely used color space for image display. It is composed of three color components *red*, *green*, and *blue*. These components are called "*additive primaries*" since a color in RGB space is produced by adding them together. In contrast, CMY space is a color space primarily used for printing. The three color components are *cyan*, *magenta*, and *yellow*. These three components are called "*subtractive primaries*" since a color in CMY space is produced through light absorption. Both RGB and CMY space are device-dependent and perceptually non-uniform.

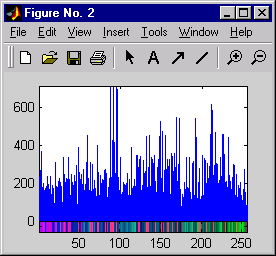
The CIE L\*a\*b\* and CIE L\*u\*v\* spaces are device independent and considered to be perceptually uniform. They consist of a luminance or *lightness* component (*L*) and two *chromatic* components *a* and *b* or *u* and *v*. CIE L\*a\*b\* is designed to deal with subtractive colorant mixtures, while CIE L\*u\*v\* is designed to deal with additive colorant mixtures. In HSV (or HSL, or HSB) space is widely used in computer graphics and is a more intuitive way of describing color. The three color components are *hue*, *saturation* (lightness) and *value* (*brightness*). The hue is invariant to the changes in illumination and camera direction and hence more suited to object retrieval. RGB coordinates can be easily translated to the HSV (or HLS, or HSB) coordinates by a simple formula [27].

The opponent color space uses the opponent color axes (*R-G*, *2B-R-G*, *R+G+B*).This representation has the advantage of isolating the brightness information on the third axis. With this solution, the first two chromaticity axes, which are invariant to the changes in illumination intensity and shadows, can be down-sampled since humans are more sensitive to brightness than they are to chromatic information.

## Methods of Representation

The main method of representing color information of images in CBIR systems is through color histograms. A color histogram is a type of bar graph, where each bar represents a particular color of the color space being used. In MatLab for example you can get a color histogram of an image in the RGB or HSV color space. The bars in a color histogram are referred to as bins and they represent the x-axis. The number of bins depends on the number of colors there are in an image. The y-axis denotes the number of pixels there are in each bin. In other words how many pixels in an image are of a particular color.

An example of a color histogram in the HSV color space can be seen with the following image:

To view a histogram numerically one has to look at the color map or the numeric representation of each bin.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Colour Map***  ***(x-axis)*** | | |  | ***Number of Pixels per Bin***  ***(y-axis)*** |
| H | ***S*** | ***V*** |  |
| 0.9922 | 0.9882 | 0.9961 |  | 106 |
| 0.9569 | 0.9569 | 0.9882 |  | 242 |
| 0.9725 | 0.9647 | 0.9765 |  | 273 |
| 0.9176 | 0.9137 | 0.9569 |  | 372 |
| 0.9098 | 0.8980 | 0.9176 |  | 185 |
| 0.9569 | 0.9255 | 0.9412 |  | 204 |
| 0.9020 | 0.8627 | 0.8980 |  | 135 |
| 0.9020 | 0.8431 | 0.8510 |  | 166 |
| 0.9098 | 0.8196 | 0.8078 |  | 179 |
| 0.8549 | 0.8510 | 0.8941 |  | 188 |
| 0.8235 | 0.8235 | 0.8941 |  | 241 |
| 0.8471 | 0.8353 | 0.8549 |  | 104 |
| 0.8353 | 0.7961 | 0.8392 |  | 198 |
| . | . | . |  | . |
| . | . | . |  | . |
| . | . | . |  | . |

As one can see from the color map each row represents the color of a bin. The row is composed of the three coordinates of the color space. The first coordinate represents hue, the second saturation, and the third, value, thereby giving HSV. The percentages of each of these coordinates are what make up the color of a bin. Also one can see the corresponding pixel numbers for each bin, which are denoted by the blue lines in the histogram.

## Quadratic Distance Metric

**The equation we used in deriving the distance between two color histograms is the quadratic distance metric:**



The equation consists of three terms. The derivation of each of these terms will be explained in the following sections. The **first term** consists of the difference between two color histograms; or more precisely the difference in the number of pixels in each bin. This term is obviously a vector since it consists of one row. The number of columns in this vector is the number of bins in a histogram. The **third ter**m is the transpose of that vector. The **middle term** is the similarity matrix. The **final result** **d** represents the color distance between two images. The closer the distance is to zero the closer the images are in color similarity. The further the distance from zero the less similar the images are in color similarity.

**Histograms**

**Quantization** in terms of color histograms refers to the process of reducing the number of bins by taking colors that are very similar to each other and putting them in the same bin. By default the maximum number of bins one can obtain using the histogram function in MatLab is 256. For the purpose of saving time when trying to compare color histograms, one can quantize the number of bins. Obviously quantization reduces the information regarding the content of images but as was mentioned this is the tradeoff when one wants to reduce processing time.

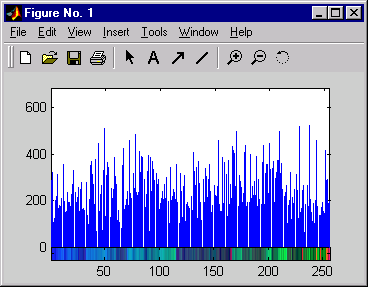
There are **two types of color histograms**, Global color histograms (***GCH***s) and Local color histograms (***LCH***s). A GCH represents one whole image with a single color histogram. An LCH divides an image into fixed blocks and takes the color histogram of each of those blocks [7]. LCHs contain more information about an image but are computationally expensive when comparing images. The GCH is the traditional method for color based image retrieval. However, it does not include information concerning the color distribution of the regions [7] of an image. Thus when comparing GCHs one might not always get a proper result in terms of similarity of images.

**We used GCHs in extracting the color features of images**. In analyzing the histograms there were a few issues that had to be dealt with. First there was the issue of how much we would quantize the number of bins in a histogram. By default the number of bins represented in an image's color histogram using the ***imhist()*** function in MatLab is 256. Meaning that in our calculations of similarity matrix and histogram difference, the processing would be computationally expensive. Initially we decided to quantize the number of bins to 20. This means that colors that are distinct yet similar are assigned to the same bin reducing the number of bins from 256 to 20. This obviously decreases the information content of images, but decreases the time in calculating the color distance between two histograms. On the other hand keeping the number of bins at 256 gives a more accurate result in terms of color distance. Later on we went back to 256 bins due to some inconsistencies obtained in the color distances between images. This had nothing to do with quantizing the image but rather with the types of images we were using which will be further elaborated later on in the Results section.

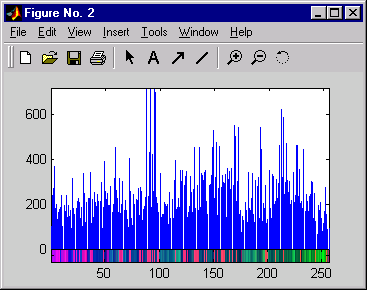
The second issue was in which color space would we present our color map. Should it be **RGB** or **HSV**? This was solved right away when we found that ***QBIC***'s similarity matrix equation was using the **HSV** color space in its calculation. There hasn't been any evidence to show which color space generates the best retrieval results, thus the use of this color space did not restrict us an anyway.

## Similarity Matrix

As can be seen from the color histograms of two images ***Q*** and ***I*** in the figure below, the color patterns observed in the color bar are totally different. This is further confirmed when one sees the respective color maps in the following table…



(a) Image Q



(a) Image I

Figure: Color Histograms of two images.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Color Map of image Q*** | | |  | ***Color Map of image I*** | | |
| 0.9608 | 0.8980 | 0.7843 |  | 0.9922 | 0.9882 | 0.9961 |
| 0.9373 | 0.9059 | 0.8235 |  | 0.9569 | 0.9569 | 0.9882 |
| 0.9098 | 0.8510 | 0.7765 |  | 0.9725 | 0.9647 | 0.9765 |
| 0.9255 | 0.8588 | 0.8039 |  | 0.9176 | 0.9137 | 0.9569 |
| 0.8627 | 0.8275 | 0.7961 |  | 0.9098 | 0.8980 | 0.9176 |
| 0.9098 | 0.8431 | 0.7216 |  | 0.9569 | 0.9255 | 0.9412 |
| 0.9137 | 0.8392 | 0.6627 |  | 0.9020 | 0.8627 | 0.8980 |
| 0.9059 | 0.7882 | 0.6510 |  | 0.9020 | 0.8431 | 0.8510 |
| 0.9451 | 0.8275 | 0.6824 |  | 0.9098 | 0.8196 | 0.8078 |
| 0.9569 | 0.7882 | 0.5922 |  | 0.8549 | 0.8510 | 0.8941 |
| 0.9137 | 0.7765 | 0.5961 |  | 0.8235 | 0.8235 | 0.8941 |
| 0.9412 | 0.7961 | 0.5569 |  | 0.8471 | 0.8353 | 0.8549 |
| 0.8471 | 0.7843 | 0.7176 |  | 0.8353 | 0.7961 | 0.8392 |
| 0.8275 | 0.7843 | 0.6745 |  | 0.8431 | 0.7804 | 0.7843 |
| 0.9020 | 0.8392 | 0.6667 |  | 0.7961 | 0.7804 | 0.8353 |
| 0.8980 | 0.7333 | 0.5843 |  | 0.7882 | 0.7725 | 0.7882 |
| 0.9020 | 0.7216 | 0.5333 |  | 0.8235 | 0.8314 | 0.8118 |
| . | . | . |  | . | . | . |
| . | . | . |  | . | . | . |
| . | . | . |  | . | . | . |

###### Table: Colour Maps of two images.

A simple distance metric involving the subtraction of the number of pixels in the 1st bin of one histogram from the 1st bin of another histogram and so on is not adequate. This metric is referred to as a *Minkowski-Form Distance Metric*, shown below, which only compares the “same bins between color histograms [3]”.

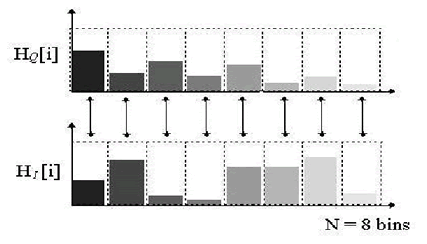


Figure: Minkowski Distance Approach…

The Minkowski distance treats all bins of the feature histogram entirely independently and does not account for the fact that certain pairs of bins correspond to features which are perceptually more similar than the other pairs.

This is the main reason for using the quadratic distance metric. More precisely it is the middle term of the equation or similarity matrix ***A*** that helps us overcome the problem of different color maps. The similarity matrix is obtained through a complex algorithm:

which basically compares one color bin of ***HQ*** with all those of ***HI*** to try and find out which color bin is the most similar, as shown below:

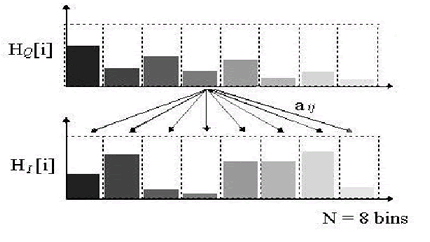


Figure: Quadratic Distance Approach…

This is continued until we have compared all the colour bins of ***HQ***. In doing so we get an ***N* x *N*** matrix, ***N*** representing the number of bins. What indicates whether the color patterns of two histograms are similar is the diagonal of the matrix, shown below. If the diagonal entirely consists of ones then the color patterns are identical. The further the numbers in the diagonal are from one, the less similar the color patterns are. Thus the problem of comparing totally unrelated bins is solved.

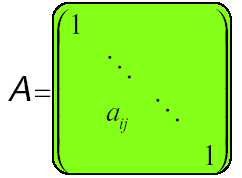


Figure: Similarity Matrix A, with a diagonal of ones…[3]

It has been shown that quadratic form distance can lead to perceptually more desirable results than Euclidean distance and histogram intersection method as it considers the cross similarity between colors.

**Texture**

In the field of computer vision and image processing there is no clear-cut definition of texture. This is because available texture definitions are based on texture analysis methods and the features extracted from the image. However texture can be thought of as repeated patterns of pixels over a spatial domain, of which the addition of noise to the patterns and their repetition frequencies result in textures that can appear to be random and unstructured. Texture properties are the visual patterns in an image that have properties of homogeneity that do not result from the presence of only a single color or intensity. The different texture properties as perceived by the human eye are regularity, directionality, smoothness and coarseness

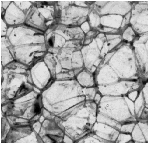
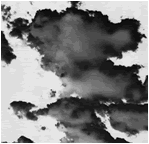
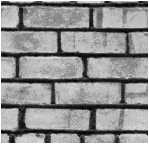
Image textures have useful applications in image processing and computer vision. They include

• Recognition of image regions using texture properties, otherwise known as texture classification.

• Recognition of texture boundaries using texture properties, otherwise known as texture segmentation.

• Texture synthesis, generation of texture images from known texture models.

• Extraction of image shape using texture properties.



***(a) Clouds***

***(b) Bricks***

***(c) Rocks***

Texture properties include:

* Coarseness
* Contrast
* Directionality
* Line-likeness
* Regularity
* Roughness

Figure: Examples of Textures…

## Texture feature extraction

Since there is no accepted mathematical definition for texture, many different methods for computing texture features have been proposed over the years. Unfortunately, there is still no single method that works best with all types of textures. The commonly used methods for texture feature description are statistical and transform-based methods [**8**], [**9**].

#### Statistical method

Statistical methods analyses the spatial distribution of grey values by computing local features at each point in the image, and deriving a set of statistics from the distribution of the local features.

For optimum classification purposes, what concern us are the statistical techniques of characterization. This is because it is these techniques that result in computing texture properties. The most popular statistical representations of texture are:

* Co-occurrence Matrix
* Tamura Texture
* Wavelet Transform

## Pyramid-Structured Wavelet Transform

Textures can be modeled as quasi-periodic patterns with spatial/frequency representation. The wavelet transform transforms the image into a multi-scale representation with both spatial and frequency characteristics. This allows for effective multi-scale image analysis with lower computational cost [10]. According to this transformation, a function, which can represent an image, a curve, a signal etc., can be described in terms of a coarse level description in addition to others with details that range from broad to narrow scales [12].

Unlike the usage of sine functions to represent signals in Fourier transforms, in wavelet transform, we use functions known as wavelets. Wavelets are finite in time, yet the average value of a wavelet is zero [10]. In a sense, a wavelet is a waveform that is bounded in both frequency and duration. While the Fourier transform converts a signal into a continuous series of sine waves, each of which is of constant frequency and amplitude and of infinite duration, most real-world signals (such as music or images) have a finite duration and abrupt changes in frequency. This accounts for the efficiency of wavelet transforms. This is because wavelet transforms convert a signal into a series of wavelets, which can be stored more efficiently due to finite time, and can be constructed with rough edges, thereby better approximating real-world signals.

Examples of wavelets are Coiflet, Morlet, Mexican Hat, Haar and Daubechies. Of these, Haar is the simplest and most widely used, while Daubechies have fractal structures and are vital for current wavelet applications [2]. These two are outlined below:

**Haar Wavelet**

The Haar wavelet family is defined as [2]:

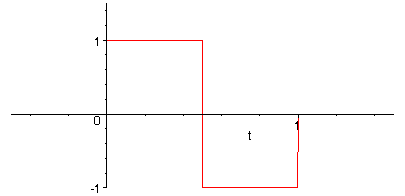


Figure: Haar Wavelet Example

***Daubechies Wavelet***

The Daubechies wavelet family is defined as [2]:

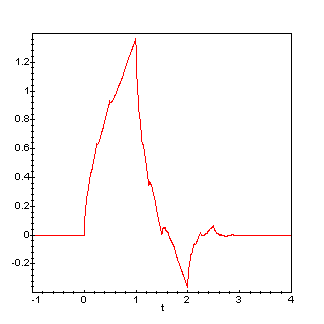


Figure: Daubechies Wavelet Example

the *wavelet transform* [21, 62] provides a multi-resolution approach to texture analysis and classification [19, 54]. Wavelet transforms decompose a signal with a family of basic functions:

obtained through translation and dilation of a mother wavelet ψ(*x*), i.e.,

where *m* and *n* are dilation and translation parameters. A signal *f*(*x*) can be represented as:

The computation of the wavelet transforms of the 2D signal involves recursive filtering and sub-sampling. At each level the signal is decomposed into four frequency sub-bands LL,LH,HL,HH where L denotes low frequency and h denotes high frequency. Two major wavelet transform types used for tecture analysis are pyramid structures wavelet transform and tree structures wavelet transform.

After the decomposition, feature vectors can be constructed using the mean and standard deviation of the energy distribution of each sub-band at each level. For three-level decomposition, PWT results in a feature vector of 3x4x2 components. For TWT, the feature will depend on how sub-bands at each level are decomposed. A fixed decomposition tree can be obtained by sequentially decomposing the LL, LH,

and HL bands, and thus results in a feature vector of 52x2 components. Note that in this example, the feature obtained by PWT can be considered as a subset of the feature obtained by TWT.

We used a method called the pyramid-structured wavelet transform for texture classification. Its name comes from the fact that it recursively decomposes sub signals in the low frequency channels. It is mostly significant for textures with dominant frequency channels. For this reason, it is mostly suitable for signals consisting of components with information concentrated in lower frequency channels [10]. Due to the innate image properties that allows for most information to exist in lower sub-bands, the pyramid-structured wavelet transform is highly sufficient.

Using the pyramid-structured wavelet transform, the texture image is decomposed into four sub images, in low-low, low-high, high-low and high-high sub-bands. At this point, the energy level of each sub-band is calculated [see energy level algorithm in next section]. This is first level decomposition. Using the low-low sub-band for further

## Energy Level

Energy Level Algorithm:

1. Decompose the image into *four* sub-images

2. Calculate the energy of all decomposed images at the same scale, using



where ***M*** and ***N*** are the dimensions of the image, and ***X*** is the intensity of the pixel located at row ***i*** and column ***j*** in the image map.

3. Repeat from step 1 for the low-low sub-band image, until index ***ind*** is equal to 5. Increment ***ind***.

Using the above algorithm, the energy levels of the sub-bands were calculated, and further decomposition of the low-low sub-band image. This is repeated five times, to reach fifth level decomposition. These energy level values are stored to be used in the Euclidean distance algorithm.

Euclidean Distance Algorithm:

1. Decompose query image.
2. Get the energies of the first dominant ***k*** channels.
3. For image ***i*** in the database obtain the ***k*** energies.
4. Calculate the Euclidean distance between the two sets of energies, using



1. Increment ***i***. Repeat from step 3.

Using the above algorithm, the query image is searched for in the image database. The Euclidean distance is calculated between the query image and every image in the database. This process is repeated until all the images in the database have been compared with the query image. Upon completion of the Euclidean distance algorithm, we have an array of Euclidean distances, which is then sorted. The five topmost images are then displayed as a result of the texture search.

**Shape defined**

Shape of an object is the characteristic surface configuration as represented by the outline or contour. Shape recognition is one of the modes through which human perception of the environment is executed. It is important in CBIR because it corresponds to region of interests in images.

In image processing, shape is the binary image consisting of contour or outline of objects, obtained after segmentation.

In CBIR system designed for specific domain such as trademarks and silhouettes of tools, shape segmentation can be automatic and effective. However this is not the case for CBIR system having heterogeneous database. In this case shape segmentation may be difficult or sometimes impossible.

Shape feature representations are categorized according to the techniques used. They are boundary-based and region-based. Boundary-based technique describes the shape region by using its external characteristics, for example pixel along the object boundary, while the region-based technique describes the shape region by using its internal characteristics, for example the pixel contained in the region.

Simple boundary-based shape descriptors include area, perimeter, compactness, eccentricity, elongation, and orientation. Complex boundary-based descriptors include Fourier descriptors, grid descriptors, chain codes and statistical moments [**23**].

**Area**

Area is the number of pixels in the region described by the shape. The real area of each pixel may be taken into consideration to get the real size of a region. In Fig. ?? each pixel has area of one square unit. The total area represented by the shape is 28 square units because the total pixel inside the shape region is 28.

**Perimeter**

Perimeter is the number of pixels in the boundary of the shape. In Fig. ?? the total number of pixels on the boundary of the shape is 32

**Compactness**

Compactness is a measure of how closely packed is the shape. The most compact shape is a circle while all other shapes have compactness larger than that of a circle (π4). Example of compact and non-compact shape is shown in Fig.??.



**Demonstration of Area as shape descriptor**



**Demonstration of perimeter as shape descriptor**



**Example of compact and non-compact shape**

**Eccentricity**

Eccentricity is the ratio of the longest chord of a shaped object to longest chord perpendicular to it. Eccentricity is a measure of how circular a shape is. For a perfectly circular shape the eccentricity is zero. Elliptical orbits have eccentricities between zero and one. Objects with different eccentricities are shown in the following figure:



Objects with different eccentricities

**Moments**

**Moment of inertia** or **second-order moments** are shape descriptor that measures the distribution of mass relative to axes through the center of gravity. It is based on the physics concept of moment of inertia. In theory distinct shapes have distinct moments.

Classical shape representation uses a set of *moment invariants*. If the object *R* is represented as a binary image, then the central moments of order *p+q* for the shape of object *R* are defined as:



where the sums are taken over all points (*x*, *y*) contained within the region. The centre of gravity of the region are



and



*n*, the total number of points contained in the region, is a measure of its area. This central moment can be normalized to be scale invariant:



From the moment a lot of useful information about the object can be obtained. For example a binary image:



The area, is given by the zeroth moment:



The center of mass is the first moment



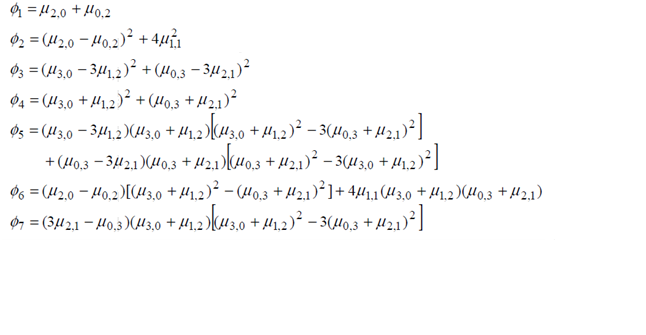
Seven moments are derived from the normalized second and third central moments. They are labelled



and are invariant to changes of position, scale and orientation [24] . The values of each of seven invariant moments provide information about the shape of the object [**25**].

φ1, φ2 are always positive. Higher φ2 values means that a shape is wider than it is tall, whereas φ2 will be higher if the shape is taller. Φ4,φ5 are measures of covariance*.* What this means is that shapes that are strongly diagonal, or skewed, will give higher values. Φ6,φ7 are measures of asymmetry. If φ6 is positive, it means that the shape is bulkier to the left and more outstretched to the right of the centroid. If φ7 is negative the shape is more outstretched upwards and downwards if it is positive.

The set of moment invariants to translation, rotation, and scale is :



### GUI

The Graphical User Interface was constructed using ***MatLab GUIDE*** or ***Graphical User Interface Design Environment***. Using the layout tools provided by ***GUIDE***, we designed the following graphical user interface figure for our application:

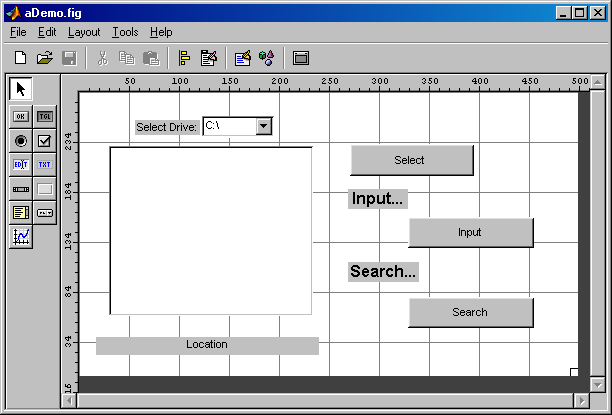


Figure: GUI Design

In addition to the above outlined design, we also designed a simple menu structure, using the ***Menu Editor***, as shown below:

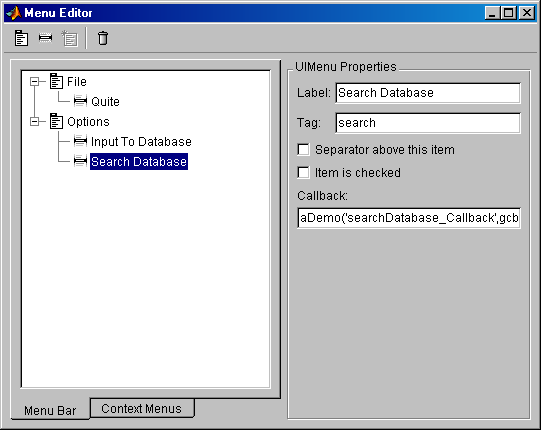


Figure: Menu Editor specification for the menu…

The above design yields the following application window on run time:

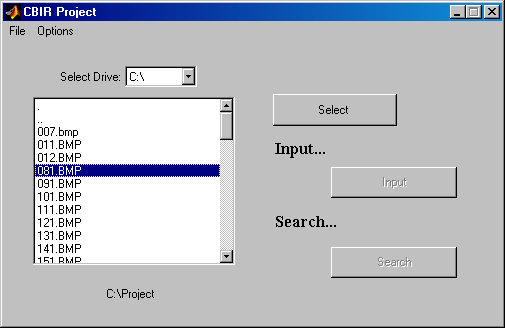


Figure: Application window at runtime…

The handlers for clicking on the buttons are coded using MatLab code to perform the necessary operations.

**RESULTS - Examples**

**Color**

After obtaining all the necessary terms, similarity matrix, and colour histogram differences, for a number of images in our database, we implemented the results in the final equation, ***Quadratic Distance Metric***. Surprisingly a number of inconsistencies kept appearing in terms of the colour distances between certain images. Images that where totally unrelated had colour distances smaller than those that where very similar. An example of this can be seen with the following three images: a mosque, a hockey game, and another picture of the same hockey game, as seen below:

***(a) faceoff3 (b) faceoff4 (c) mosque***

Figure: Tested Images…

As can be seen from the following table, the results are not consistent with how the images look to the human eye.

|  |  |
| --- | --- |
| ***Images*** | ***Colour distance between image histograms*** |
| faceoff3 vs faceoff4 | 10.77 |
| faceoff3 vs mosque | 9.99 |

###### Table: Colour Distances for Compressed Images…

This was done again and again with a number of images, and resulted in the same inconsistencies. What turned out to be the cause of all this, were the type of images we were using. At first we thought the only thing that could give inconsistent results like this was comparing images of different sizes, but we had resized all the images in our database to 256x256 before testing our algorithm. The images we had in our database where all 24-bit ***JPEG***s. The problem with ***JPEG*** images is that they are compressed and the compression algorithm seems to affect the way the histograms are derived. We found this out by converting some of the images in our database to 8-bit uncompressed bit maps. The same images that where tested in ***JPEG*** format where tested again as ***BMP***s. What resulted was consistent with how the images looked to the human eye. Images that looked similar gave small color distances compared to those that looked very different. This can be seen in the following table, which shows the same images as those in the previous table but in ***BMP*** format.

|  |  |
| --- | --- |
| ***Images*** | ***Colour distance between image histograms*** |
| faceoff3 vs faceoff4 | 4.39 |
| faceoff3 vs mosque | 6.10 |

###### Table: Colour Distances for Uncompressed Images

In realizing that our error was due to image format, we decided to convert all our images to uncompressed ***BMP***s. This obviously is not consistent with full ***CBIR systems*** available in the market, which take any type of image as a query in any format, but for the purposes of this project we did not want to delve into compression issues.

In converting all of our images to 8-bit uncompressed ***BMP***s there was a slight change in the way we dealt with their respective colour maps. What we previously did was index an image before using the ***imhist*** function. What indexing does is quantize the colour map by letting the user specify the number of bins. This obviously reduces the processing time in terms of calculating colour distances since you don't have 256 bins to compare. When loading 8-bit uncompressed images into a variable, MatLab does not let you quantize their colour maps. It gives you an error when you try to index them. Thus we where forced to stick to the default value of 256 bins for all our colour histograms.

### Example

To demonstrate the project application, we implemented the following example:

* We started the application by typing ***aDemo*** and pressing return in the MatLab Command Window. The application window started.
* In the application window, we selected the *Options* menu, and selected *Search Database*. This enabled the browsing window, to browse to a ***BMP*** file.
* Upon highlighting a ***BMP*** file, the select button became enabled. **Note:** Only 8-bit uncompressed ***BMP***s are suitable for this application. In this example, we selected the following bmp.
* The highlighted ***BMP*** is then selected by pressing the *Select* button.
* Next, pressing the *Search* button started the search.

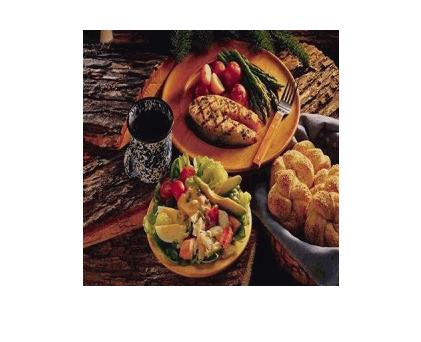


Figure: The query image: 935.bmp

## 8.5.1 Colour Extraction & Matching

Using the *colour feature extraction algorithm* described above, where the histograms of the query image and the images in the database are compared using the *Quadratic Distance Metric*, we obtained the following top 10 results:



Figure: Colour Results for the searching for 935.bmp

The above results are sorted according to the quadratic distance.

## 8.5.2 Texture Extraction & Matching

Using the *texture feature extraction algorithm* described above, where the energies of the query image and the *colour result* images’ sub-bands are compared using the ***Euclidean Distance Metric*,** we obtained the following top 5 results:

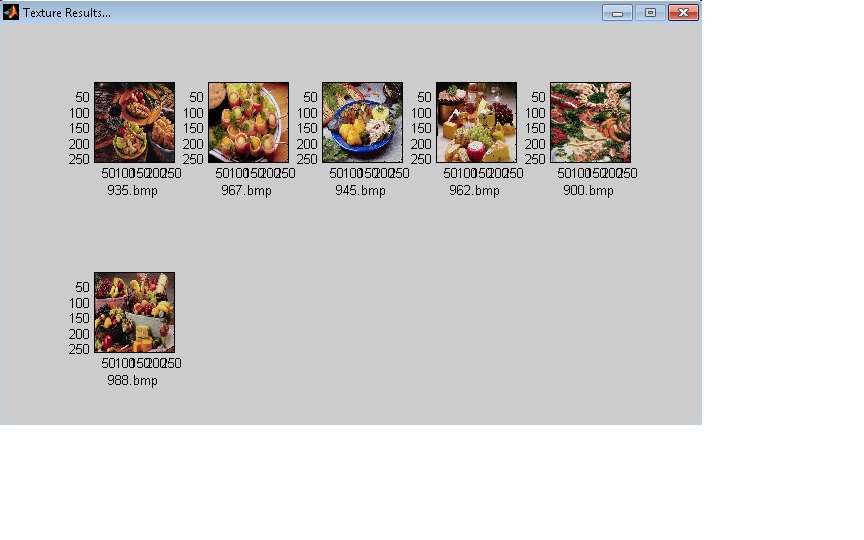
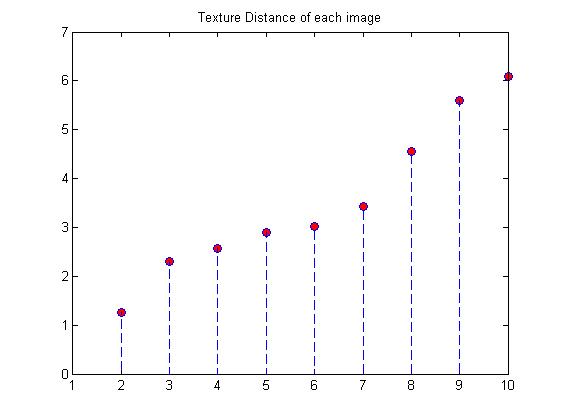


Figure: Texture Results for the searching for 935.bmp

The above results are sorted according to the Euclidean distance. These are shown below:



###### Figure: Euclidean distance between query and results…

By observing the images in our database, we can actually say that the above results represent the closest matches to the query image chosen.

**Moments Extraction & Matching**

Using the *moments feature extraction algorithm* described above, where the central moments of the query image and the *texture result* images’ are compared using the ***Euclidean Distance Metric*,** we obtained the following top 4 results:

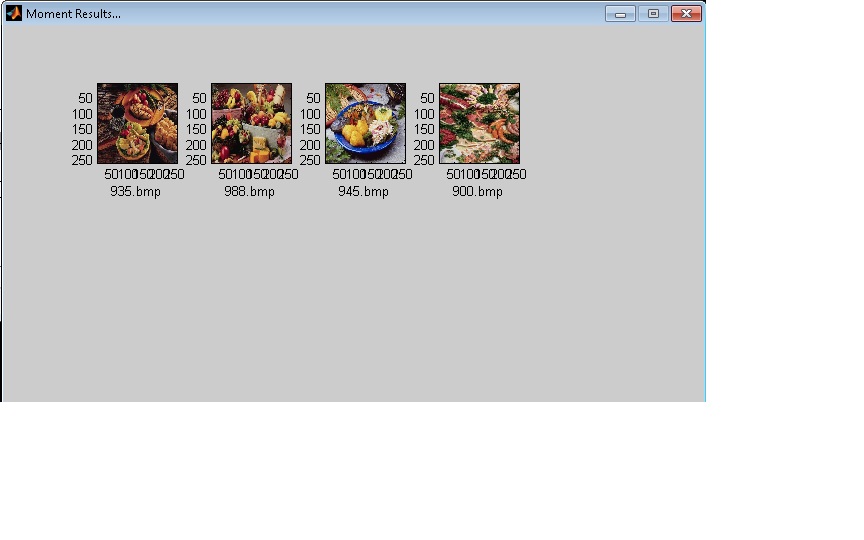
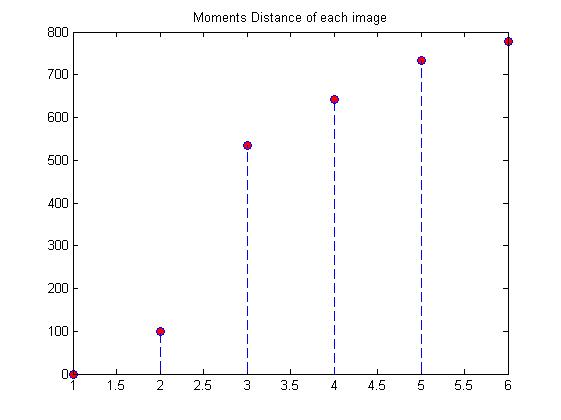
****

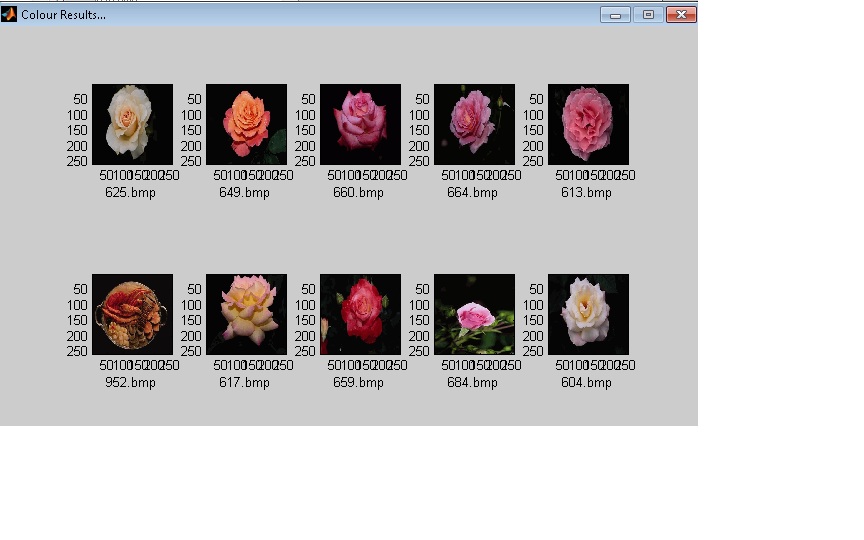
Figure: Results after applying moment invariants for 935.bmp

The above results are sorted according to the Euclidean distance. These are shown below:

****

Next we represent some other experimental results to show how our approach performs in various images.

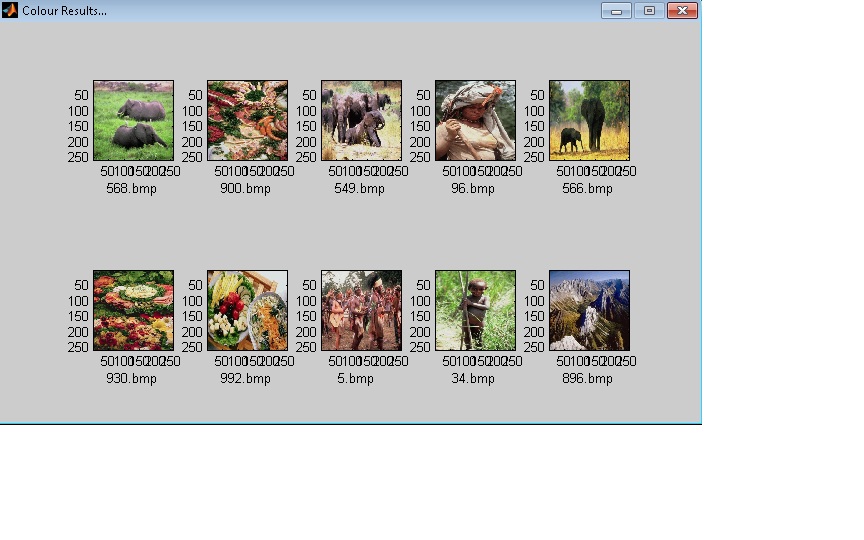
The following query image of a flower performed rather well, giving 9 out of 10 matches similar flower images.



On the contrary trying the search with a different flower image with a more”green” background confused the algorithm and gave poor results.



Querying an elephant image gives some matches, but still the color algorithm which is the first step in our approach doesn’t perform well in giving more elephant images in the top ten results, since the database consists of many other images with similar color patterns than the ones of the query that do not represent elephants.



**Concluding remarks**

The dramatic rise in the sizes of images databases has stirred the development of effective and efficient retrieval systems. The development of these systems started with retrieving images using textual connotations but later introduced image retrieval based on content. This came to be known as CBIR or Content Based Image Retrieval. Systems using CBIR retrieve images based on visual features such as color, texture and shape, as opposed to depending on image descriptions or textual indexing. In this project, we have researched various modes of representing and retrieving the image properties of color, texture and shape. Due to lack of time, we were only able to fully construct an application that retrieved image matches based on colour and texture only.

The application performs a simple color-based search in an image database for an input query image, using color histograms. It then compares the color histograms of different images using the *Quadratic Distance Equation*. Further enhancing the search, the application performs a texture-based search in the color results, using wavelet decomposition and energy level calculation. It then compares the texture features obtained using the *Euclidean Distance Equation*. A more detailed step would further enhance these texture results, using a shape-based search.

CBIR is still a developing research area. As image compression, digital image processing, and image feature extraction techniques become more developed, CBIR maintains a steady pace of development in the research field. Furthermore, the development of powerful processing power, and faster and cheaper memories contribute heavily to CBIR development. This development promises an immense range of future applications using CBIR

At the end of this review, we would like to present our view on a few trends:

1. The driving force

In our review, most of the journal contributions are from the last five years. We are aware of the fact that much of what we have said here will be outdated soon. The impetus behind content-based image retrieval is given by the wide availability of digital sensors, the Internet, and the falling price of storage devices. Given the magnitude of these driving forces, it is believed that content-based retrieval will continue to grow in every direction in the following years.

What is needed for the future is more precise foundations. For some of the reviewed papers, it was not clear what problem they were trying to solve or whether the proposed method would perform better than an alternative. A classification of usage-types, aims, and purposes would be very helpful here, including criteria for distinguishing among domain types. In spite of the difficulties, it is our belief that content-based retrieval in the end will not be part of the field of computer vision alone. The man-machine interface, domain knowledge and database technology each will have their impact on the product.

2. The heritage of computer vision

An important obstacle to overcome before content-based image retrieval could take off was to realize that image retrieval does not entail solving the general image understanding problem. It may be sufficient that a retrieval system present similar images, similar in some user-defined sense. Strong segmentation of the scene and complete feature descriptions may not be necessary at all to achieve the similarity ranking. Of course, the deeper one goes into the semantics of the pictures, the deeper the understanding of the picture will have to be, but that could very well be based on categorizing pictures rather than on a precise understanding.

3. The influence on computer vision

In reverse, content based image retrieval offers a different look at traditional computer vision problems. In the first place, content-based retrieval has brought large data sets. Where the number of test images in a typical journal paper was well under a hundred until very recently, a state-of-the-art paper in content-based retrieval reports experiments on thousands of images. Of course, the purpose is different for computer vision and content-based retrieval. It is much easier to compose a general data set of arbitrary images rather than the specific ones needed in a computer vision application, but the stage has been set for more robustness. For one thing, to process a thousand images at least demands software and computational method to be robust.

In the second place, content-based retrieval has run into the absence of a general method for strong segmentation. Especially for broad domains and for sensory conditions where clutter and occlusion are to be expected, strong segmentation into objects is hard, if not impossible. Content-based retrieval systems have dealt with the segmentation bottleneck in a few creative ways.

In the third place, content-based retrieval has revitalized interest in color image processing. This is due to superior identification of tri-valued intensities in identifying an object, as well as to the importance of color in the perception of images. As content based is user-oriented, color cannot be left out. The purpose of most image color processing here is to reduce the influence of accidental conditions of the scene and the sensing (i.e., the sensory gap) by computing sensing and scene invariant representations. Progress has been made in tailored color space representation for well-described classes of variant conditions. Also, the application of local geometrical descriptions derived from scale space theory will reveal viewpoint and scene independent salient point sets, thus opening the way to similarity of images on a small number of most informative regions or points.

Finally, attention for invariance has been revitalized as well with many new features and similarity measures. For content-based retrieval, invariance is just one side of the coin, where discriminating power is the other. Little work has been reported so far to establish the remaining discriminating power of properties. This is essential as the balance between stability against variations and retained discriminating power determines the effectiveness of a property.

4. Similarity and learning

Similarity is an interpretation of the image based on the difference between two elements or groups of elements. For each of the feature types, a different similarity measure is needed. For similarity between feature sets, special attention has gone to establishing similarity between histograms due to their computational efficiency and retrieval effectiveness. Where most attention has gone to color histograms, it is expected that histograms of local geometric properties and texture will follow.

1. The need for databases

When data sets grow in size and when larger data sets define more interesting problems, both scientifically as well as for the public, the computational aspects can no longer be ignored.

The connection between content-based image retrieval and database research is likely to increase in the future. Already, the most promising efforts are interdisciplinary, but so far, problems like the definition of suitable query languages, efficient search in high dimensional feature space, search in the presence of changing similarity measures are largely unsolved.

**References**

[2] C. Carson, M. Thomas, S. Belongie, J.M. Hellerstein, and J. Malik,

ªBlobworld: A System for Region-Based Image Indexing and

Retrieval,º Proc. Visual Information Systems, pp. 509-516, June 1999

[3] James Z. Wang, Jia Li and Gio Wiederhold, ``SIMPLIcity: Semantics-Sensitive Integrated Matching for Picture Libraries,'' **IEEE Transactions on Pattern Analysis and Machine Intelligence,** vol. 23, no. 9, pp. 947-963, 2001.

[4] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom

et al. ªQuery by Image and Video Content: The QBIC System,º

IEEE Computer, vol. 28, no. 9, 1995.

[5]. John P. Eakins and Margaret E. Graham, Content-based image retrieval, a report to the JISC technology applications programme**,** *Institute for image database research, University of Northumbria at Newcastle, U.K,* January 1999.

[6]. Hideyuki Tamura and Naokazu Yokoya. Image Database Systems: A Survey**,** *PatternRecognition*, 17(1):29–49, 1984

[7] Shengjiu Wang, “A Robust CBIR Approach Using Local Color Histograms,” Department of Computer Science, University of Alberta, Edmonton, Alberta, Canada, Tech. Rep. TR 01-13, October 2001, Found at:

<http://citeseer.nj.nec.com/wang01robust.html>

[8]. Zijun Yang and Jay Kuo, Survey on content-based analysis, indexing and retrieval techniques and status report of MPEG-7,*Tamkang journal of science and engineering, vol.2, No.3 pp.101-118, 1999*

[9]. T. Ojala and M. Pietikäinen, Texture Classfication*, Machine vision and Media signal Processing Unit*, University of Oulu, Finland

[10] Sharmin Siddique, “A Wavelet Based Technique for Analysis and Classification of Texture Images,” Carleton University, Ottawa, Canada, Proj. Rep. 70.593, April 2002.

[11] W.Y. Ma and B. Manjunath, ªNaTra: A Toolbox for Navigating

Large Image Databases,º Proc. IEEE Int'l Conf. Image Processing,

pp. 568-571, 1997.

[12] G. D. Magoulas, S. A. Karkanis, D. A. Karras and M. N. Vrahatis, “Comparison Study of Textural Descriptors for Training Neural Network Classifiers”, in Proceedings of the 3rd IEEE-IMACS World Multi-conference on Circuits, Systems, Communications and Computers, vol. 1, 6221-6226, Athens, Greece, July 1999, Found at:

<http://www.brunel.ac.uk/~csstgdm/622.pdf>

[18] Y. Rubner, L.J. Guibas, and C. Tomasi, ªThe Earth Mover's

Distance, Multi-Dimensional Scaling, and Color-Based Image

Retrieval,º Proc. DARPA Image Understanding Workshop, pp. 661-

668, May 1997.

[20] J. Shi and J. Malik, ªNormalized Cuts and Image Segmentation,º

Proc. Computer Vision and Pattern Recognition, pp. 731-737, June

1997.

[21] J.R. Smith and S.-F. Chang, ªVisualSEEk: A Fully Automated

Content-Based Image Query System,º Proc. ACM Multimedia,

pp. 87-98, Nov. 1996.

[22] J.R. Smith and C.S. Li, ªImage Classification and Querying Using

Composite Region Templates,º Int'l J. Computer Vision and Image

Understanding, vol. 75, nos. 1-2, pp. 165-174, 1999.

[23] Dengsheng Zhang, and Guojun Lu, Content-based shape retrieval using different shape descriptors *Grippsland school of computing and information technology, Monash university*, Churchill, Victoria, Australia.

[24]. Rafael C. Gonzales and Richaerds E. Woods, *Digital image processing* second edition, 2002

[25]. Jean-Marc Pelletier, Computer vision documentation,www.iamas.ac.jp/~jovan02/cv/jit\_cv\_doc.pdf

[27] J.Z. Wang, J. Li, R.M. Gray, and G. Wiederhold, ªUnsupervised

Multiresolution Segmentation for Images with Low Depth of

Field,º IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23,

no. 1, pp. 85-91, Jan. 2001.

[28] J.Z. Wang, G. Wiederhold, O. Firschein, and X.W. Sha, ªContent-

Based Image Indexing and Searching Using Daubechies' Wavelets,

º Int'l J. Digital Libraries, vol. 1, no. 4, pp. 311-328, 1998.

[29] G. Ciocca and R. Schettini, ªUsing a Relevance Feedback

Mechanism to Improve Content-Based Image Retrieval,º Proc.

Visual '99: Information and Information Systems, pp. 107-114, 1999.

[88] T. Kato, T. Kurita, N. Otsu, and K. Hirata, ªA Sketch Retrieval

Method for Full Color Image DatabaseÐQuery by Visual

Example,º Proc. ICPR, Computer Vision and Applications, pp. 530-

533, 1992.

[68] A. Hiroike, Y. Musha, A. Sugimoto, and Y. Mori, ªVisualization of

Information Spaces to Retrieve and Browse Image Data,º Proc.

Visual '99: Information and Information Systems, pp. 155-162, 1999.

[168] M.J. Swain, ªSearching for Multimedia on the World Wide Web,º

Proc. Int'l Conf. Multimedia Computing and Systems, pp. 33-37, 1999.

[31] I.J. Cox, M.L. Miller, T.P. Minka, and T.V. Papathomas, ªThe

Bayesian Image Retrieval System, PicHunter: Theory, Implementation,

and Pychophysical Experiments,º IEEE Trans. Image

Processing, vol. 9, no. 1, pp. 20-37, 2000.

[48] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom,

M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker,

ªQuery by Image and Video Content: The QBIC System,º IEEE

Computer, 1995.

[170] D.J. Swets and J. Weng, ªHierarchical Discriminant Analysis for

Image Retrieval,º IEEE Trans. Pattern Analysis and Machine

Intelligence, vol. 21, no. 5, pp. 386-401, May 1999.

[79] A.K. Jain and A. Vailaya, ªShape-Based Retrieval: A Case Study

with Trademark Image Databases,º Pattern Recognition, vol. 31,

no. 9, pp. 1,369-1,390, 1998.

[74] B. Huet and E.R. Hancock, ªLine Pattern Retrieval Using

Relational Histograms,º IEEE Trans. Pattern Analysis and Machine

Intelligence, vol. 21, no. 12, pp. 1,363-1,371, Dec. 1999.