CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

The aim of our project is to design a desktop application using which object based and image based searching from an image can be performed by the user.

This application is developed for desktop computers only. It provides simple interface so that it can be used by naïve users also. The searching is to be performed based on 3 factors i.e. color, shape and texture feature of an object. The system also provide user with an interface i.e. setting option so that he can retrieve most similar results by adjusting weightage factor for each features & make some changes in size & numbers of query as well as resulting images according to his/her preferences.

1.2 OVERVIEW

With the evolution of computing world, initially there was demand for designing fast and efficient processors. Later, there was demand for designing cheap and energy efficient device. As the time grew, various processors have been designed to achieve these goals. After certain time, now we have sufficiently powerful and cheap computer hardware available. With the advent of internet, the amount of information grew exponentially with time. At this time, we have extremely large amount of information. So, now there is a need for designing an intelligent computer system. A system, which does not only follows the algorithm but also evolves its behavior with time and recognizes some kind of knowledge present in the given massive information. This is called Smart Computing era. One most important application of such needs was to design an intelligent search engine.

Image search engines are mainly designed to find the most appropriate image as per user’s requirement from a collection of large amount of image. Most web based image search engines rely purely on metadata and this produces a lot of garbage in the results. Also having humans manually enter keywords for each & every images in a large database can be inefficient, expensive, and impractical and may not capture every keyword that describes the image. Thus a system that can filter images based on their content would provide better indexing and return more accurate results. Hence CBIR is desirable.

Content-based image retrieval (CBIR), also known content-based visual information retrieval (CBVIR) is the application of computer vision techniques in the process of retrieving desired images from a large collection of the images based of syntactical image features. The term Content-Based Image Retrieval (CBIR) originated in 1992, when it was used by T. Kato to describe experiments into automatic retrieval of images from a database, based on the colors and shapes present. ‘Content-based’ means that the search engine will analyze the actual contents of the image rather than the metadata such as keywords, tags and descriptions associated with the image. The term 'content' in this context might refer to colors, shapes, textures, or recognizing an entire object present in the image.

Traditional image search engines were based on metadata i.e. keyword, tag, description of an image. Our search engine searches images based on visual information inside image i.e. 3 features including color, texture & shape. Our proposed system is Desktop-based image search engine in which user can store all the images anywhere on local computer according to their convenience and use our search engine to search the related images from all the images.

The basic processing required for retrieving content of images are:-

1. Image Segmentation
2. Image Feature Extraction
3. Feature Representation and Storing in some database.
4. Mapping Features to semantics
5. Storing and Indexing images based on these features
6. Image similarity measures and retrieval.

We have used following different algorithms:-

1. Segmentation algorithm – Statistical Region Merging Algorithm (SRMA)
2. Color feature representation – Hue, Saturation & Value i.e. HSV histogram
3. Texture feature representation – Gray Level Co-occurrence Matrix (GLCM)
4. Shape feature representation – Distance Auto-Correlogram (DAC)

Important Modules in our Project are as follows:–

1. Loader
2. Database
3. Image to feature vector convertor (IFV)
4. Comparator
5. User interface(GUI)

Our system provides following options to users –

1. Search based on Image
2. Search based on Object

Developing a CBIR system has lots of applications such as fingerprint identification, biodiversity information systems, digital libraries, crime prevention, medicine, historical research, image clustering, and image mining, and personalization, image similarity identification etc.

1.3 EXISTING SYSTEM

As a part of study of existing systems we have studied some of systems built around the idea of content based image retrieval. Using these existing systems we have got clear idea about what necessary changes we should do in our proposed system to make it useful and better system. Here we are listing below some of the important existing systems. Following are the existing system in detail.

4.1 BLOBWORLD
4.2 PATSEEK
4.3 PHOTOBOOK
4.4 QBIC
4.5 SIMBA

4.1. BLOBWORLD [2]

This system was developed by Computer Science Division, University of California, Berkeley.. Blobworld is a system for image retrieval based on finding coherent image regions which roughly correspond to objects. Each image is automatically segmented into regions (“blobs") with associated color and texture descriptors. Querying is based on the attributes of one or two regions of interest, rather than a description of the entire image. In order to make large-scale retrieval feasible, we index the blob descriptions using a tree. It is distinct from color-layout matching as in QBIC in that it is designed to find objects or parts of objects; each image is treated as an ensemble of a few “blobs" representing image regions which are roughly homogeneous with respect to color and texture. Each blob is described by its color distribution and mean texture descriptors.

The features used for querying are the color, texture, location, and shape of regions (blobs) and of the background. The color is described by a histogram of 218 bins of the color coordinates in Lab-space. Texture is represented by mean contrast and anisotropy over the region, as the 2D coordinate. Shape is represented by (approximate) area, eccentricity, and orientation.

The user first selects a category, which already limits the search space. In an initial image, the user selects a region (blob), and indicates the importance of the blob. Next, the user indicates the importance of the blob's color, texture, location, and shape. More than one region can be used for querying.

To match two color histograms h_1 and h_2, the quadratic form distance is used:

$d(h_1, h_2)=(h_1-h_2)^T A(h_1-h_2)$ [9], where $A=(a_{ij})$ is a symmetric matrix of weights representing the similarity between color bins i and j. The distance between two texture descriptors is the Euclidean distance between their coordinates in representation space. The distance between centroids is the Euclidean distance. The distances are combined into a single final distance. It uses R*-trees index structures for data representable as points in an N-dimensional space .Shorter paths from root to leaf in an index tree lead to fewer disk accesses to reach the leaves and thus faster data retrieval.

Limitations
a. The system was designed only for image archival systems. Hence if the image organization changes then entire image has to be processed again increasing additional overhead to the system. Thus, it is not suitable for desktop users where there are frequent changes in the image database.

b. The similarity of two images is appearance based phenomenon. Hence two images much similar according to one user may not be similar according other user’s perception. Thus, ideal image retrieval system should be designed in such a way to identify user’s preference and change the similarity matching criteria accordingly. Blobworld system has not been designed to take user’s perspective into consideration. Hence the retrieved result may not best for each user.

4.2. PATSEEK (CBIR for US Pattern Database) [3]

PATSEEK System was developed as a Content Based Image Search Engine for US based patent System. It was developed by *Avinash Tiwari, Veena Bansal*at IIT Kanpur for American Patent System. In this system, user can check the similarity of his query image with the images that exist in US patents. Here, user can specify a set of key words that must exist in the text of the patents whose images will be searched for similarity. PATSEEK automatically grabs images from the US patent database on the request of the user and represents them through an edge orientation auto-correlogram (EOAC). [3]

The gradient of edges is used to quantize edges into 36 bins of 5 degrees each. The edge orientation auto-correlogram is then formed which is a matrix, consisting of 36 rows and 4 columns. Each element of this matrix indicates the number of edges with similar orientation. Columns 1, 2, 3 and 4 give the number of edges that are 1, 3, 5 and 7 pixels apart. Each row corresponds to 5 degrees bin.

One of major issue was to extract different images from a page. A page image may contain more than one individual image. To extract the individual images from these page images, connected components or blocks in an image are identified. To identify a particular block, horizontal and vertical scan to image are performed. The thresholds are set to 1 mm. A block that is less than 5 square cm is discarded as noise. The edges that have less than 10 percent of the maximum possible intensity are ignored from further consideration. Feature Retrieval method called the EOAC which generate 144 real numbers which corresponds to feature of an image. The magnitude and gradient of the edges is calculated using *Canny Edge Operator*. For comparison purpose, two edges with k pixel distance apart are said to be similar if the absolute values of their orientations and amplitude differences are less than an angle and an amplitude threshold value, respectively.

For practical implementation of the system, *Java (Sun JDK 1.4.1) was used*. The feature vector database was initially created on *Oracle* and later on moved to *MySQL version 12.21*.[8] For querying the system, user can interact with system and provide the query image and can also specify angle of rotation (0 to 180) of the resultant image. The angle of rotation helps to determine similar images even if object image is rotated by some degrees.

To demonstrate working of this project it was implemented for some sample images. The experimental database contains approximately 200 images that have been picked up from the patent database of United States Patent Office. For the performance evaluation, 15 different images from collection of images have been selected. For each query image, a set of relevant images in the database have been manually identified. An ideal image retrieval system is expected to retrieve all the relevant images. One of the popular measures is *recall rate* that is the ratio of number of relevant images retrieved and total number of relevant images in the collection. The *precision rate* is the ratio of the number of relevant images retrieved and total number of images in the collection. The *precision rate* is the ratio of the number of relevant images retrieved and total number of images in the collection. A recall rate of 100% for 61% of query images and an average 32% recall rate for rest of the images were observed. The precision rate varied between 10% and 35%.

Limitations

a. The system does not take color or texture into consideration. It takes only shape feature into consideration. This puts a limit on system functionality. Thus image similar in color content will not be taken into account. Hence usability of system is limited only to black and white images.

b. The performance of system is poor in terms of response time since user has to wait for more time to get the result. Thus, the usability reduces.

4.3. PHOTOBOOK [4] [15]

Photobook is a computer system that allows the user to browse larger image database quickly and efficiently using both text annotation information in an AI database and by having the computer search the images directly based on their content. This allows people to search in flexible and intuitive manner using semantic categories and analogies.

Photobook implements three different approaches to constructing image representations for querying purposes, each for a specific type of image content: faces, 2D shapes and texture images. The first two representations are similar in the way that they offer a description relative to an average of a few prototypes by using the eigenvectors of a covariance matrix as an orthogonal coordinate system of the image space.

First a pre-processing step is done in order to normalize the input image for position, scale and orientation. Given a set of training images, $\Gamma_1, \Gamma_2, \dots \Gamma_M$ where Γ_{1i} is anxn array of intensity values, their variation from the average, $\psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i$ is given by $\varphi_i = \Gamma_{i-}\psi$. This set of vectors is then subjected to the *Karhunen-Loève expansion*, the result being a set of *M* eigenvectors u_k and Eigen values λ_k of the covariance matrix. $C = \frac{1}{M}\sum_{i=1}^{M}\varphi_i * \varphi_i^T$ In representing a new image region, Γ, only $M_1 < M$ eigenvectors with the largest eigenvalues are used, thus the point in the Eigen image space corresponding to the new image is
$\Omega = (\omega_1, \omega_2, \dots \omega_{M_1})$, where $\omega_i = u_k^T(\Gamma - \psi)$, k = 1,…, $M_1 < M$.

In a texture description, an image is viewed as a homogeneous 2D discrete random field, which by means of a Wold decomposition, is expressed as the sum of three orthogonal components. These components correspond to periodicity, directionality and randomness. In creating a shape description, first a silhouette is extracted and a number of feature points on this are chosen (such as corners and high-curvature points). This feature points are then used as nodes in building a finite element model of the shape. Solving the following eigenvalue problem$K\varphi_i = w_i^2 . M\varphi_i$, where *M* and *K* are the mass and stiffness matrices, respectively, the modes of the model are computed. These are the eigenvectors,φ_i, which are next used for determining a feature point correspondence between this new shape and some average shape.

To perform a query, the user selects some images from the grid of still images displayed and/or enters an annotation filter. From the images displayed, the user can select another query images and reiterate the search. The distance between two Eigen image representations, Ω_1 andΩ_2, is$\in_{ij}^2 = ||\Omega_i - \Omega_j||^2$. Two shapes are compared by calculating the amount of strain energy needed to deform one shape to match the other.

Prior to any database search, a few prototypes that span the image category are selected. For any image in the database, its distance to the average of the prototypes is computed and stored for future database search. At query time, the distance of the query image to the average is computed and the database is reordered according to this. Images in the database are sorted by similarity with the query images and presented to the user page by page.

Limitations
a. The system provides very complex tool (textual annotations) for query, which is not easily understandable by a normal user which has no idea about query syntax.

b. Since it is based on textual annotation if a user fires a query like “bunch of tree”, then system does not give desire result.

c. It works efficiently only for human facial recognition, which is generally gray scale images, it is not efficient for color and an edge image as more processing is required.

d. It uses compression algorithm for preprocessing which is a lossy compression.

e. Ideally, the system should show two objects same even if they are rotated. Thus the representation should be rotation, shape deformation invariant. In this system, this care is not taken. Hence if an objects image is taken from different angle then it will not necessary show them similar. Thus true negative ratio increases in retrieve result and the result quality is poor.

4.4. QBIC (Query by Image Content) [5] [16]

One of the first content based image retrieval systems available was the QBIC system (Query by Image Content) from IBM. The QBIC system uses three types of features.[10]

- **a.** A color histograms to describe the color distribution,
- **b.** A moment based shape feature to describe shapes,

c. A texture descriptor based on contrast, coarseness, and directionality to account for textures.

Color features computed are: the 3D average color vector of an object or the whole image in RGB, YIQ, Lab, and Munsell color space and a 256-dimensional RGB color histogram. If *x* is an *n*-dimensional color histogram and $C = [c_1 c_2 \ldots c_n]$ is a $3 X N$ matrix whose columns represent the RGB values of the *n* quantized colors, the average color vector x_{avg} is C_x. The texture features used in QBIC are modified versions of the coarseness, contrast, and directionality features proposed by Tamura. The shape features consist of shape area, circularity, eccentricity, major axis orientation and a set of algebraic moment invariants. The major axis orientation and the eccentricity are computed from the second order covariance matrix of the boundary pixels: the major axis orientation as the direction of the largest eigenvector and eccentricity as the ratio of the smallest eigenvalue to the largest one. For the database images, these shape features are extracted for all the object contours, semi automatically computed in the database population step. In this process, the user enters an approximate object outline, which is automatically aligned with the nearby image edges, using the active contours technique. In this object identification step, the user can also associate text to the outlined objects. The 18 algebraic moment invariants are the eigenvalues of the matrices $M_{[2,2]}$,$M_{[2,3]} \text{ X } M_{[3,2]}$, $M_{[3,3]}$,$M_{[3,4]} \text{ X } M_{[4,3]}$, $M_{[4,4]}$, $M_{[4,5]} \text{ X } M_{[5,4]}$ where the elements of $M_{[i,j]}$ are scaled factors of the central moments. QBIC also implemented a method of retrieving images based on a rough user sketch. For this purpose, images in the database are represented by a reduced binary map of edge points. This is obtained as follows: first, the color image is converted to a single band luminance; using a *CannyEdge Detector*, the binary edge image is computed and is next reduced to size*64 X 64*. Finally this reduced image is thinned.

QBIC allows queries based on example images, user-constructed sketches or/and selected color and texture patterns. In the last case, the user chooses colors or textures from a sampler. The percentage of a desired color in an image is adjusted by moving sliders.

For the average color, the distance between a query object and database object is a weighted Euclidean distance, where the weights are the inverse standard deviation for each component over the samples in the database. In matching two color histograms, two distance measures are used: one low-dimensional, easy to compute (the average color distance) and one much more computationally expensive (the quadratic histogram distance). The first one (which is computed for all the images in the database) acts as a filter, limiting the expensive matching computation to the small set of images retrieved by the first matching. The average color distance is $d^2_{avg}(x,y)=(x_{avg}-y_{avg})^t.(x_{avg}-y_{avg})$. The histogram quadratic distance is given by $d^2_{hist}(x,y)=(x-y)^t.A(x-y)$, where the symmetric color similarity matrix *A* is given by $a_{ij} = 1-d_{ij}/d_{max}$, with d_{ij} being the L_2 distance between the colors *i* and *j* in the RGB space and $d_{max} = max_{i,j}d_{ij}$[23]. The texture distance is a weighted Euclidean distance, with the weighting factors being the inverse variances for each of the three texture components over the entire database. Two shapes are matched also by a similar weighted Euclidean distance between shape feature vectors. In a query by sketch, after reducing the binary sketch image drawn by the user to size 64 X 64, a correlation based matching is performed, a kind of template matching. This is done by partitioning the user sketch into 8 X 8blocks of 8 X 8pixels and find the maximum correlation of each block of the sketch within a search area of 16 X 16pixels in the image database (this is done by shifting the 8 X

8block in the search area). This local correlation score is computed on the pixel level using logical operations. The matching score of a database image is the sum of the correlation scores of all local blocks.

Limitations

- **a.** The system is designed with an assumption that all the images in the database are noise free. Hence this puts a limitation of system as a general purpose search engine. Noisy data present in image increases the true negative and false positive ratio in the images and hence irrelevant images may also be considered as relevant and relevant images may be classified as irrelevant.

- **b.** Memory requirement is very high. Thus, for a very large database it is practically impossible to implement on user's desktop computer.

4.5. SIMBA (Search IMages By Appearance) [6] [12] [17]

SIMBA is content based image retrieval system performing queries based on image appearance. Based on general construction method(integration over the transformation group) it drives invariant feature histograms that catch different cues on image content features that are strongly influenced by color and textual features that are robust to illumination change. By a weighted combination of these features here user can adapt the similarity measure according to his need, thus improves the retrieval result.

The system uses a histogram of a function *f* of transformed local pixel patterns of image *I*(*x*, *y*), taken over all rigid motions *g*. For example, *f*(*I*) could be I(0, 1)* I(2, 0), and the histograms could be constructed of the values$g_{t_0,t_1,\varphi}(I(0,1).g_{t_0,t_1,\varphi}(I(2,0),t_0 = 0 \ldots n,\ \ t_1 = 0, \ldots m$, $\varphi = 0$. This is based on the observation that the integration over all transformations *g* (Haar integral) of *f*(*g*(*I*)) makes *f* invariant under these transformations. The histogram is used rather than the integral, to keep more local information. By combining different functions *f* and different color layers (RGB in the online demo, however, also HSV, luminance and chromaticity's are supported) a multidimensional histogram is built. A fuzzy version of the histogram was developed to get rid of the discontinuities of the histogram operator at bin boundaries. The evaluation of the features is speeded up by Monte Carlo estimation instead of a full calculation.

The user is presented with a selectable number (1-25) of random images from the database. By selecting one of these, a query for similar images is started. Alternatively, the user can specify a URL for his own query image, whose size must be comparable to the size of the database images (384x256).The results of a query are shown in decreasing order of match row by row. The user may specify resolution and number of rows/columns for the results, before starting a query. The result display can be used for a new query by clicking on one of the displayed images (to come iteratively closer to the desired image type) or by specifying a new URL.

Limitations

a. SIMBA works on only two features one is color and other one is brightness, for texture it uses illumination change, which does not work if edges are present. So it is not efficient for texture based search.

b. It is efficient only for 2500 images. If Image database increases more than 2500 the efficiency decreases.

c. It works for MPEG-7 image. It does not take all formats of images.

1.4 SCOPE OF PROJECT

1. Desktop Search Engine

Our project is for desktop users where users can interact with system easily. User can store all the images anywhere on local computer according to their convenience and use our search engine to search the related images from all the images.

2. Feature Representation

For content based processing, image contents need to be represented. Contents of images are represented by their color, texture and shape. Together these values can be used to identify each object uniquely from the image. Hence from image segmentation, each object is identified and later its descriptions are represented using its features. These features are obtained from different image format sources.

3. Search based on Image

In our system, user can input the image as a query from any folder which he has to search. The system will search for that particular image and will output all the similar type of images.

4. Search based on Objects

User can select query image and choose to query by search by object. Then system will return the different objects present inside that image using segmentation. User will then search again based on that object & system will return all the images containing that particular object or related to that object from the image database.

5. User Friendly

Our system also provides settings option to the user in which user can set as per his convenience. In this option we are providing user with Similarity criteria for all 3 features so that user can vary their dimensions using slider. Also it provides search results settings like maximum number of resulting search images, number of images per row. Another option is it also generates log file. User can also set height & width of query image & resulting image as per choice.

1.5 TEAM STRUCTURE

The team structure that we used here is Democratic Decentralized. The features of such a team structure is as follows:

There is no center of authority, which allows free flow of ideas among the group members. This free flow is required since the number of group members is limited to just three.
It is democratic in the sense that decisions are taken by forming a general consensus on an issue.

In a way democratic and decentralized go hand in hand as the team is decentralized in the sense that decisions are not dependent on any central authority and this leads to democracy as decisions are therefore taken by collaboration.

CHAPTER 2

PROJECT ANALYSIS

2.1 CONTENT BASED IMAGE RETRIVAL

An **image retrieval** system is a computer system for browsing, searching and retrieving images from a large database of digital images. Most traditional and common methods of image retrieval utilize some method of adding metadata such as captioning, keywords, or descriptions to the images so that retrieval can be performed over the annotation words.

Content-based image retrieval (**CBIR**), also known as **query by image content** (**QBIC**) and **content-based visual information retrieval** (**CBVIR**) is the application of computer vision techniques to the image retrieval problem, that is, the problem of searching for digital images in large databases [1].

"Content-based" means that the search will analyze the actual contents of the image rather than the metadata such as keywords, tags, and/or descriptions associated with the image. The term 'content' in this context might refer to colors, shapes, textures, or any other information that can be derived from the image itself. CBIR is desirable because most web based image search engines rely purely on metadata and this produces a lot of garbage in the results. Also having humans manually enter keywords for images in a large database can be inefficient, expensive and may not capture every keyword that describes the image. Thus a system that can filter images based on their content would provide better indexing and return more accurate results.

2.1.1 History [1]

The term Content-Based Image Retrieval (CBIR) was originated in 1992, when it was used by T. Kato to describe experiments into automatic retrieval of images from a database, based on the colors and shapes present. Since then, the term has been used to describe the process of retrieving desired images from a large collection on the basis of syntactical image features. The techniques, tools and algorithms that are used originate from fields such as statistics, pattern recognition, signal processing, and computer vision.

2.1.2 CBIR Techniques [1]

Many CBIR systems have been developed, but the problem of retrieving images on the basis of their pixel content remains largely unsolved.

Query Techniques [1]

Different implementations of CBIR make use of different types of user queries.

2.1.2.1 Query by example [1]

Query by example is a query technique that involves providing the CBIR system with an example image that it will then base its search upon. The underlying search algorithms may vary depending on the application, but result images should all share common elements with the provided example. Options for providing example images to the system include:

- A preexisting image may be supplied by the user or chosen from a random set.
- The user draws a rough approximation of the image they are looking for, for example with blobs of color or general shapes
- This query technique removes the difficulties that can arise when trying to describe images with words.

2.1.2.2 Semantic retrieval [1]

The ideal CBIR system from a user perspective would involve what is referred to as *semantic* retrieval, where the user makes a request like "find pictures of dogs" or even "find pictures of Abraham Lincoln". This type of open-ended task is very difficult for computers to perform - pictures of chihuahuas and Great Danes look very different, and Lincoln may not always be facing the camera or in the same pose. Current CBIR systems therefore generally make use of lower-level features like texture, color, and shape; although some systems take advantage of very common higher-level features like faces. Not every CBIR system is generic. Some systems are designed for a specific domain, e.g. shape matching can be used for finding parts inside a CAD-CAM database.

2.1.2.3 Other query methods [1]

Other query methods include browsing for example images, navigating customized/hierarchical categories, querying by image region (rather than the entire image), querying by multiple example images, querying by visual sketch, querying by direct specification of image features, and multimodal queries (e.g. combining touch, voice, etc.).

CBIR systems can also make use of *relevance feedback*, where the user progressively refines the search results by marking images in the results as "relevant", "not relevant", or "neutral" to the search query, then repeating the search with the new information.

2.1.3 Content comparison using image distance measures [1]

The most common method for comparing two images in content based image retrieval (typically an example image and an image from the database) is using an image distance measure. An image distance measure compares the similarity of two images in various dimensions such as color, texture, shape, and others. For example, a distance of 0 signifies an exact match with the query, with respect to the dimensions that were considered. As one may intuitively gather, a value greater than 0 indicates various degrees of similarities between the images. Search results then can be sorted based on their distance to query image.

2.2 OBJECT DETECTION

Object Detection plays a major role in extraction of an object from a given image. It is performed by segmentation of the image. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics.

The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image (see edge detection). Each of the pixels in a region is similar with respect to some characteristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s). When applied to a stack of images, typical in Medical imaging, the resulting contours after image segmentation can be used to create 3D reconstructions with the help of interpolation algorithms like marching cubes.

The entire efficiency of object feature extraction depends on how better segmentation is performed. If the segmentation is very accurate then results obtained will always be better and same.

Statistical Region Merging Algorithm [9]

For object detection from an image, we have used statistical region merging algorithm. It uses disjoint set data structure algorithm to construct segments. It performs texture based segmentation of the image. Following are steps for SRM:-

Initially each pixel of the image is assigned a different value. Two pixels with same value of membership belong to same segment and two pixels with different pixel values represent different segments. Different iterations are performed. In each iteration,

Two adjacent pixels with same or high similarity (greater than threshold value) are merged with the idea that they both belong to same object and their parent values are set to the one with dominant pixel numbers. While merging, the pixel intensity of fewer occurrences one is replaced by pixel intensity of more frequent occurring pixel. This step assures that two adjacent segments will never be assigned same color. If they are then they both belong to same object itself.
The above step is repeated for number of iterations. And for each iteration, neighboring value is increased.

From different test cases, we came to conclusion that the optimum results are obtained for 256 iterations and in each iteration the neighboring distance value is incremented by 1.

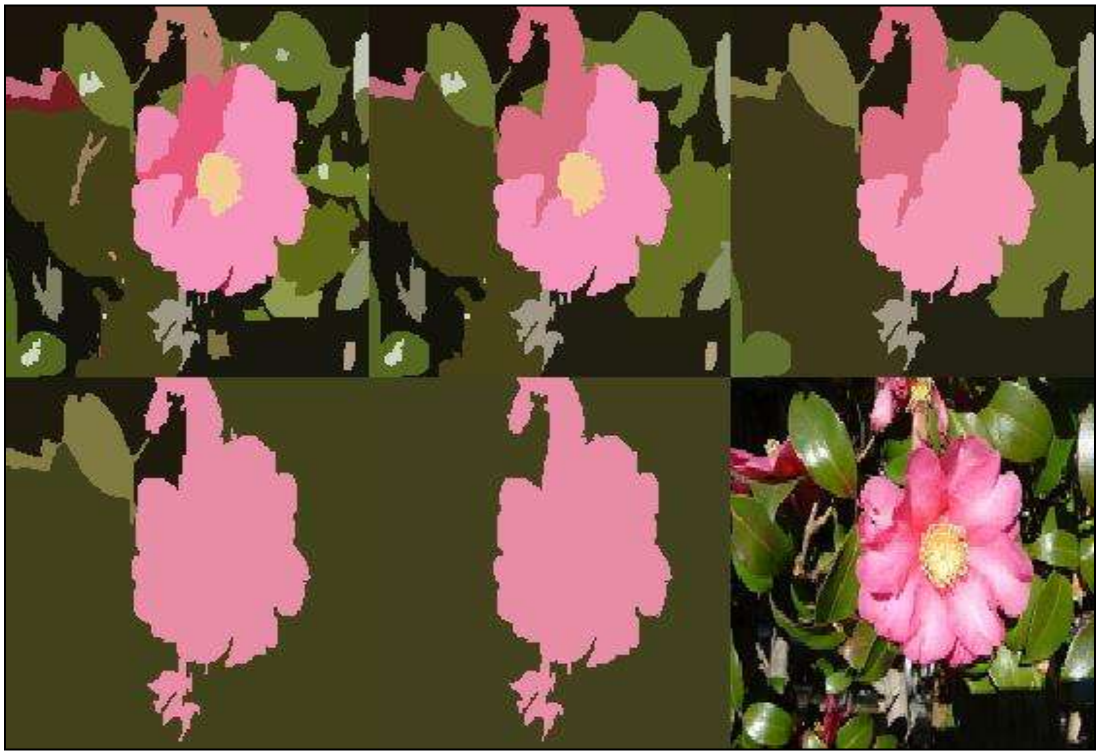


Figure 1: SRM algorithm for different iterations

2.3 COLOR REPRESENTATION

A **color model** is an abstract mathematical model describing the way colors can be represented as tuples of numbers, typically as three or four values or color components. When this model is associated with a precise description of how the components are to be interpreted (viewing conditions, etc.), the resulting set of colors is called color space. Various Color representation models exist which are used to color of an object. Following are different representation models:

2.3.1 Tristimulus Color Space [10]

This is represented by a space as a region in three-dimensional Euclidean space if one identifies the *x*, *y*, and *z* axes with the stimuli for the long-wavelength (*L*), medium-wavelength (*M*), and short-wavelength (*S*) receptors. The origin, (*S*,*M*,*L*) = (0,0,0), corresponds to black. White has no definite position in this diagram; rather it is defined according to the color temperature or white balance as desired or as available from ambient lighting.

The most saturated colors are located at the outer rim of the region, with brighter colors farther removed from the origin. As far as the responses of the receptors in the eye are concerned, there is no such thing as "brown" or "gray" light. The latter color names refer to orange and white light respectively, with an intensity that is lower than the light from surrounding areas. One can observe this by watching the screen of an overhead projector during a meeting: one sees black lettering on a white background, even though the "black" has in fact not become darker than the white screen on which it is projected before the projector was turned on. The "black" areas have not actually become darker but appear "black" relative to the higher intensity "white" projected onto the screen around it.

The human tristimulus space has the property that additive mixing of colors corresponds to the adding of vectors in this space. This makes it easy to, for example, describe the possible colors (gamut) that can be constructed from the red, green, and blue primaries in a computer display.

2.3.2 CIEXYZ Color Space [10]

One of the first mathematically defined color spaces is the CIE XYZ color space (also known as CIE 1931 color space), created by the International Commission on Illumination in 1931. These data were measured for human observers and a 2-degree field of view. In 1964, supplemental data for a 10-degree field of view were published.

Sometimes XYZ colors are represented by the luminance, Y, and chromaticity coordinates *x* and *y*, defined by:

$$x = \frac{X}{X+Y+Z}$$
$$y = \frac{Y}{X+Y+Z}$$

Mathematically, *x* and *y* are projective coordinates and the colors of the chromaticity diagram occupy a region of the real projective plane. Because the CIE sensitivity curves have equal areas

under the curves, light with a flat energy spectrum corresponds to the point (x, y) = (0.333,0.333).
The values for X, Y, and Z are obtained by integrating the product of the spectrum of a light beam and the published color-matching functions.

2.3.3 RGB [10]

Media that transmit light (such as television) use additive color mixing with primary colors of red, green, and blue, each of which stimulates one of the three types of the eye's color receptors with as little stimulation as possible of the other two. This is called "RGB" color space. Mixtures of light of these primary colors cover a large part of the human color space and thus produce a large part of human color experiences.

2.3.3.1 HSV and HSL representations [10]

Recognizing that the geometry of the RGB model is poorly aligned with the color-making attributes recognized by human vision, computer graphics researchers developed two alternate representations of RGB, HSV and HSL (*h*ue, *s*aturation, *v*alue and *h*ue, *s*aturation, *l*ightness), in the late 1970s. HSV and HSL improve on the color cube representation of RGB by arranging colors of each hue in a radial slice, around a central axis of neutral colors which ranges from black at the bottom to white at the top. The fully saturated colors of each hue then lie in a circle, a color wheel.

HSV models itself on paint mixture, with its saturation and value dimensions resembling mixtures of a brightly colored paint with, respectively, white and black. HSL tries to resemble more perceptual color models such as NCS or Munsell. It places the fully saturated colors in a circle of lightness ½, so that lightness 1 always implies white, and lightness 0 always implies black.

HSV and HSL are both widely used in computer graphics, particularly as color pickers in image editing software. The mathematical transformation from RGB to HSV or HSL could be computed in real time, even on computers of the 1970s, and there is an easy-to-understand mapping between colors in either of these spaces and their manifestation on a physical RGB device.

2.3.4 CMYK Model [10]

It is possible to achieve a large range of colors seen by humans by combining cyan, magenta, and yellow transparent dyes/inks on a white substrate. These are the *subtractive* primary colors. Often a fourth black is added to improve reproduction of some dark colors. This is called "CMY" or "CMYK" color space.

The cyan ink absorbs red light but transmits green and blue, the magenta ink absorbs green light but transmits red and blue, and the yellow ink absorbs blue light but transmits red and green. The white substrate reflects the transmitted light back to the viewer. Because in practice the CMY inks suitable for printing also reflect a little bit of color, making a deep and neutral black impossible, the K (black ink) component, usually printed last, is needed to compensate for their deficiencies. The dyes used in traditional color photographic prints and slides are much more perfectly transparent, so a K component is normally not needed or used in those media.

2.3.5 Color Representation Algorithm [10]

For color representation we have used HSV color histogram which produces 11X11X11 3dimensional matrix. Initially color is converted from RGB to HSV color space and then for each pixel value in the HSV space its frequency is counted. Since each pixel value in HSV space is from 0 to 1 only. Hence, each value multiplied by 10 and incremented by 1 will always be in the range of 1 to 11. Hence, the result generated has as specified dimension.

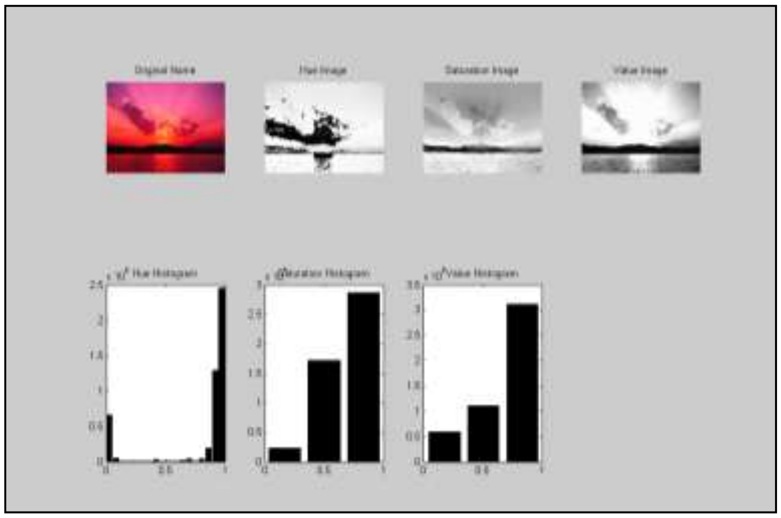


Figure 2: HSV color representation

2.4 TEXTURE REPRESENTATION

Texture is a very interesting image feature that has been used for characterization of images, a major characteristic of texture is the repetition of a pattern or patterns over a region in an image. The elements of patterns are sometimes called textons. The size, shape, color and orientation of the textons can vary over the region. The difference between two textures can be in the degree of variation of the textons. It can also be due to spatial statistical distribution of the textons in the image. Texture is an innate property of virtually all surfaces, such as bricks, fabrics, woods, papers, carpets, clouds, trees, lands and skin. It contains important information regarding underlying structural arrangement of the surfaces in an image.

Texture is generally considered as a property of a surface. It has three components: **lay**, **surface roughness**, and **waviness** [11].

2.4.1 Lay [11]

Lay is a measure of the direction of the predominant machining pattern. A *lay pattern* is a repetitive impression created on the surface of a part. It is often representative of a specific manufacturing operation. The lay may be specified when it has an effect on the function of the part. Unless otherwise specified, roughness is measured perpendicular to the lay.

2.4.2 Surface Roughness [11]

Surface roughness, more commonly shortened to *roughness*, is a measure of the finely spaced surface irregularities. In engineering, this is what is usually meant by "surface finish".

2.4.3 Waviness [11]

Waviness is the measure of surface irregularities with spacing greater than that of surface roughness. These usually occur due to warping, vibrations, or defection during machining.

2.4.4 Texture Representation [11]

Texture property of a system cannot be measured simply by histogram or some formula; it can be interpreted by evaluating various values like, rate of change of color or pixel pattern, energy content etc. It can also be calculated based on certain attributes like energy, entropy, Difference Moment, Correlation etc. Also, texture value is orientation dependent. Hence, generally for more accurate calculation, texture value is calculated from different orientation.

2.4.5 Texture Representation Algorithm [11]

For Texture representation, GLCM algorithm has been used which operates only on gray scale image. Hence query image need to be converted to gray scale. GLCM stands for **Gray Level Co-occurrence matrix**. Thus, as the name suggests, the output generated is a matrix structure. The GLCM is a tabulation of how often different combinations of pixel brightness values occur in an image. It considers the relationship between groups of two (using neighboring) pixels in the original image. GLCM texture considers the relation between two pixels at a time, called the reference and the neighbor pixel. Each pixel within the window becomes the reference pixel in turn, starting in the upper left corner and proceeding to the lower right. Pixels along the right edge have no right hand neighbor, so they are not used for this count.

2.5 SHAPE REPRESENTATION

Shape of an object located in some space is a geometrical description of the part of that space occupied by the object, as determined by its external boundary – abstracting from location and orientation in space. Shape can be represented by two ways i.e. either in terms of region or in terms of contour. Region based representation use various parameter such as area, eccentricity, major axis, minor axis Geometric Moments etc. Contour based representation on the other hand can be either based on B-spline, smallest covering polygon or chain code or in terms perimeter.

One of major thing which shape representation should cover is the spatial relationship between all the edges present at the perimeter. Parameters like area, eccentricity can be same for two different shaped bodies also and hence spatial relationship represents a unique body.

Also, shape of a body is dependent on the angle of view. Similar body if rotated by some angle might give different parameters for eccentricity, major axis, moments etc. Hence, for more accurate representation, basic transformation representation should be independent of it.

Shape Representation Algorithm [12]
The algorithm of generating DAC consists of five steps as follows:

a) *Edge detection:* The Sobel operator is more accurate about locating edges than other edge detectors. Therefore it can be used for edge detection.
b) *Computing centroidal distance matrix D(x, y):* A gray image *f* (*x*, *y*) become binary image *B*(*x*, *y*) after edge detection. First, the centroid C(x_c, y_c)is computed from the sample points as shown below.

$$x_c = \frac{1}{N}\sum_{i=0}^{N-1} x_i \, , y_c = \frac{1}{N}\sum_{i=0}^{N-1} y_i \qquad (1)$$

Then, the distance between a sample point b_i (x_i, y_i) and the centroid C(x_c, y_c)is computed as follows.

$$d(b_i, c) = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} \qquad (2)$$

Centroidal distance matrix *D*(*x*, *y*) is formed by (1) and (2).

c) *Centroidal distance standardization:* Centroidal distance matrix is a discrete, irregular number set, so it is sensitive to scaling. In order to make calculation simple and eliminate the impact of scaling, we plan all centroidal distances to the intervals between 0 to$^{x}/_{2}$by (3) as follows.[13]

$$norm\_dis = \frac{dis - dis_{\min}}{dis_{\max} - dis_{\min}} \times \frac{W}{2} \qquad (3)$$

where, *dis* are values in distance matrix, $dis_{\max}$ is the maximum centroidal distance and *dis* minis the minimum centroidal distance. norm_dist is the normalized centroidal distance for dist . W is the maximum size of contour image *B*(*x*, *y*)'s rectangular bounding box. If the size of contour image is *B*(*x*, *y*) is rectangular bounding box is a×*b*, then, W = max (*a*, *b*).

d) *Quantizing centroidal distance to n bins:* Centroidal distances are quantized to *n* bins by unit$^{W}/_{n}$.

Computing elements of DAC: In the final stage, the DAC is constructed by counting the numbers of distances belong to *n* and apart to itself 1,3,5,7 pixel unit. In order to reduce the computational complexity, 4- neighboring method has been to measure the distance, namely, only the horizontal and vertical distances are computed.

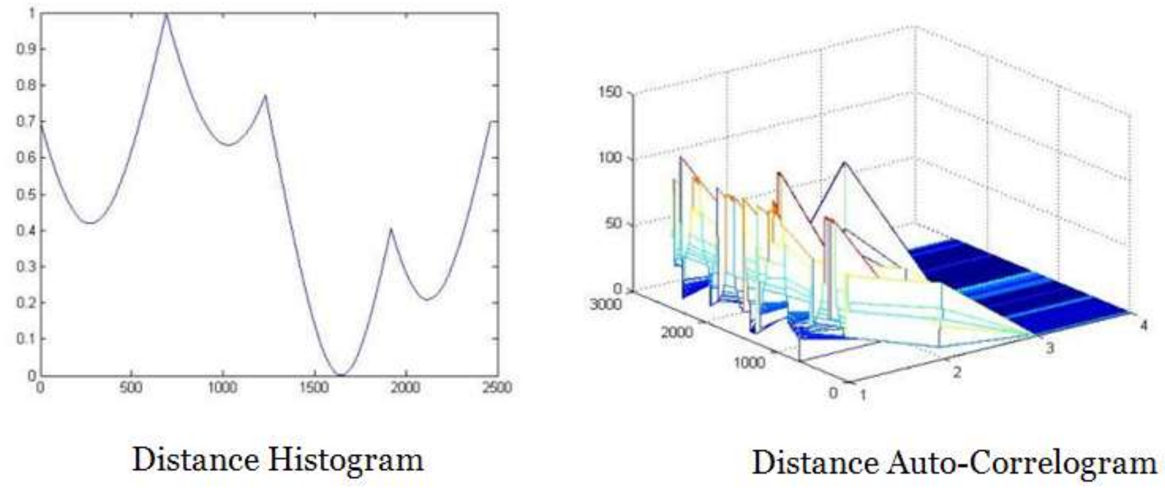


Figure 3: Distance Auto-Correlogram

2.6 IMAGE HASHING

A **hash function** is any well-defined procedure or mathematical function that converts a large, possibly variable-sized amount of data into a small datum, usually a single integer that may serve as an index to an array. The values returned by a hash function are called **hash value**.

One can use a hash function to map each record to an index into a table *T*, and collect in each bucket *T*[*i*] a list of the numbers of all records with the same hash value *i*. Once the table is complete, any two duplicate records will end up in the same bucket. The duplicates can then be found by scanning every bucket *T*[*i*] which contains two or more members, fetching those records, and comparing them. With a table of appropriate size, this method is likely to be much faster than any alternative approach (such as sorting the file and comparing all consecutive pairs). If hash key size is sufficiently large then chances of two records mapping to same bucket reduces and for very large key size, it is almost impossible for two keys to be mapped to same value. Using the same concept if key size is 128 bit then around $2^{128 =}$ 3.402 $*10^{38}$ different key values are possible. Thus chances of two records getting same key is almost negligible. Hence it can be used as unique identifier for content of record. We have used MD-5 algorithm which takes an image as input and produces 128 bit hash value. Thus basic transformations like image renaming, moving from one directory to others won't affect the key value and hence will remain unchanged. Thus, using same, image can be tracked.

MD-5 Algorithm

MD5 Algorithm for calculating signature is as follows: [14]

```
var int[64] r, k

//r specifies the per-round shift amounts
r[ 0..15] := {7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22}
r[16..31] := {5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20}
r[32..47] := {4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23}
r[48..63] := {6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21}

//Use binary integer part of the sines of integers (Radians) as constants:
for i from 0 to 63
    k[i] := floor(abs(sin(i + 1)) × (2 pow 32))

//Initialize variables:
var int h0 := 0x67452301
var int h1 := 0xEFCDAB89
var int h2 := 0x98BADCFE
var int h3 := 0x10325476

//Pre-processing:
append "1" bit to message
append "0" bits until message length in bits ≡ 448 (mod 512)
append bit /* bit (not byte) length of unpadded message as 64-bit little-
endian integer to message

//Process the message in successive 512-bit chunks:
for each 512-bit chunk of message
    break chunk into sixteen 32-bit little-endian words w[j], 0 ≤ j ≤ 15
    //Initialize hash value for this chunk:
    var int a := h0
    var int b := h1
    var int c := h2
    var int d := h3

    //Main loop:
    for i from 0 to 63
        if 0 ≤ i ≤ 15 then
            f := (b and c) or ((not b) and d)
            g := i
        else if 16 ≤ i ≤ 31
            f := (d and b) or ((not d) and c)
            g := (5×i + 1) mod 16
        else if 32 ≤ i ≤ 47
            f := b xor c xor d
            g := (3×i + 5) mod 16
        else if 48 ≤ i ≤ 63
            f := c xor (b or (not d))
            g := (7×i) mod 16
        temp := d
        d := c
        c := b
        b := b + leftrotate((a + f + k[i] + w[g]) , r[i])
        a := temp

    //Add this chunk's hash to result so far:
    h0 := h0 + a
    h1 := h1 + b
    h2 := h2 + c
    h3 := h3 + d

var char digest[16] := h0 append h1 append h2 append h3 //(expressed as
little-endian)

  //leftrotate function definition
  leftrotate (x, c)
      return (x << c) or (x >> (32-c));
```

2.7 SOFTWARE PROCESS MODEL

Process model chosen for software development indicates the phases through which software goes during its development phase and how each phase affects other process or how a particular process is carried out. Since our project is image processing based application, any change in the requirement would affect the entire developed product and product will have to be rebuilt again.

Thus, due to complexity in the nature of application, we have used the Waterfall model. It is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation and Maintenance phase.

Our project is developed using following steps followed in order:

1. Conception
2. Requirements specification (Requirements analysis)
3. Software Design
4. Implementation and Integration
5. Testing
6. Maintenance

1. Conception

Before the product is actually developed, its idea needs to be developed. Our study on various image processing applications and various search engines lead us to the idea of developing a new search engine for desktop users which can be used by the naïve users also and will lead to better search results than what existing system does.

2. Requirement Specification

Once the idea is developed, requirements need to be gathered to develop overall system. Initially we have gathered the information requirements from desktop users who are going to be ultimate user of the application. Here we got lots of ideas regarding how the system should be and what all functionality it should provide to end user. One of major challenge was working with contradictory requirements from different users. For example, more expert users wanted advance functionality whereas naïve users wanted simple to use interface. Hence working with such requirements was major challenge to us. Apart from this, we also collected information regarding how user wants to search images to be in terms of how the similarity should be measured. We also collected many more functional and non-functional requirements for the project.

3. Software Design

Initially a top level design of the software was developed showing major modules of the system and their interaction. This diagram represents architectural view of the system. In this diagram, there are five major modules i.e. Loader, Database, Image to Feature Vector Convertor (IFVC), Comparator and User Interface. Architecture diagram has been explained in detail in chapter 3.

After Architecture diagram was prepared, each component was designed based on its functional and nonfunctional behavior to fulfill certain requirements. All these modules were designed with a set of interfaces so that changes in one module won't change other module. Thus, loose coupling among inter-module communication has been used.

4. Implementation and Integration

Once each module and component of each module has been designed, development i.e. coding of modules was started. We have used Matlab R2010a for image processing and interface development using Java. Image Processing involves feature extraction of images, loading of images into database, feature comparison. All these modules were developed in Matlab and were tested for different test cases and found to be error free. After successful testing on different test cases, they were wrapped up into Jar file and were interfaced with UI developed in Java.

5. Testing

Since, different modules were written in Matlab, hence for different test cases, different set of images were chosen. After each module is developed its unit testing is performed by taking wide variety of test cases. If some unexpected result occurs, code is debugged to find source of error which is rectified and then module is tested again. Thus, unit testing helped to check functionality of each module. After unit testing, individual modules were merged to make one large system. We have used bottom up integration as it was easy to perform in our context. During integration phase, the integrated module was checked as black box to see if the functionality generated is same as expected. And finally we tested overall system to check whether it is satisfying the entire requirement mentioned earlier.

6. Maintenance

During the execution process we have found some bugs. We have fixed it. In future we hope there will not be any faults.

2.8 FEASIBILITY STUDY

Economic Feasibility

1. For the Developers

For our project we require Image Processing Toolbox (Matlab) which is free for the student on *www.mathworks.com*. We also require JDK version 1.6 along with Netbeans higher than version 5.5 for designing good GUI. Netbeans is free IDE which can be downloaded from *www.netbeans.org.* We are using Matlab for image processing (Feature extraction) and Netbeans for GUI. So it is totally free from developer side.

2. For the Users

User needs to have Java Runtime Environment (JRE) version 6 which is freely available on www.java.com and Matlab Component Runtime for executing Matlab codes which is absolutely free. As the software is distributed free of charge, no other additional expenses are there for the user.

Operational Feasibility

Since our system has simple interface, so a normal user can understand the system easily. There is less navigation in the interface. On clicking search button, system automatically searches most relevant images from database.

2.9 SYSTEM REQUIREMENTS

The following describes the various hardware and software requirements for our project on game development for mobile devices exploiting mobile technology:

Hardware Requirements

- x86 system /PC
- Any Intel or AMD x86 processor supporting SSE2 instruction set. The System should have minimum following configuration.
    - 512 MB RAM
    - 2 GB free Hard Disk space
    - 1.5 GHz Pentium or higher processor.
- The System should have recommended following configuration.
    - 1 GB RAM
    - 2 GB free hard Disk space
    - 2.0 GHz or higher Pentium Dual Core processor

Software Requirements

- Operating System: Window XP or Higher version or Ubuntu 9.10, Red Hat Enterprise Linux 5.x, SUSE Linux Enterprise Desktop 11.x, Debian 5.x
- Java Runtime Environment (JRE) version 6
- Image Processing Toolbox (from www.mathworks.com) version 6 or Higher
- Matlab Component Runtime MCR version 7

CHAPTER 3
DESIGN

3.1 ARCHITECTURE OF FELIS [13]

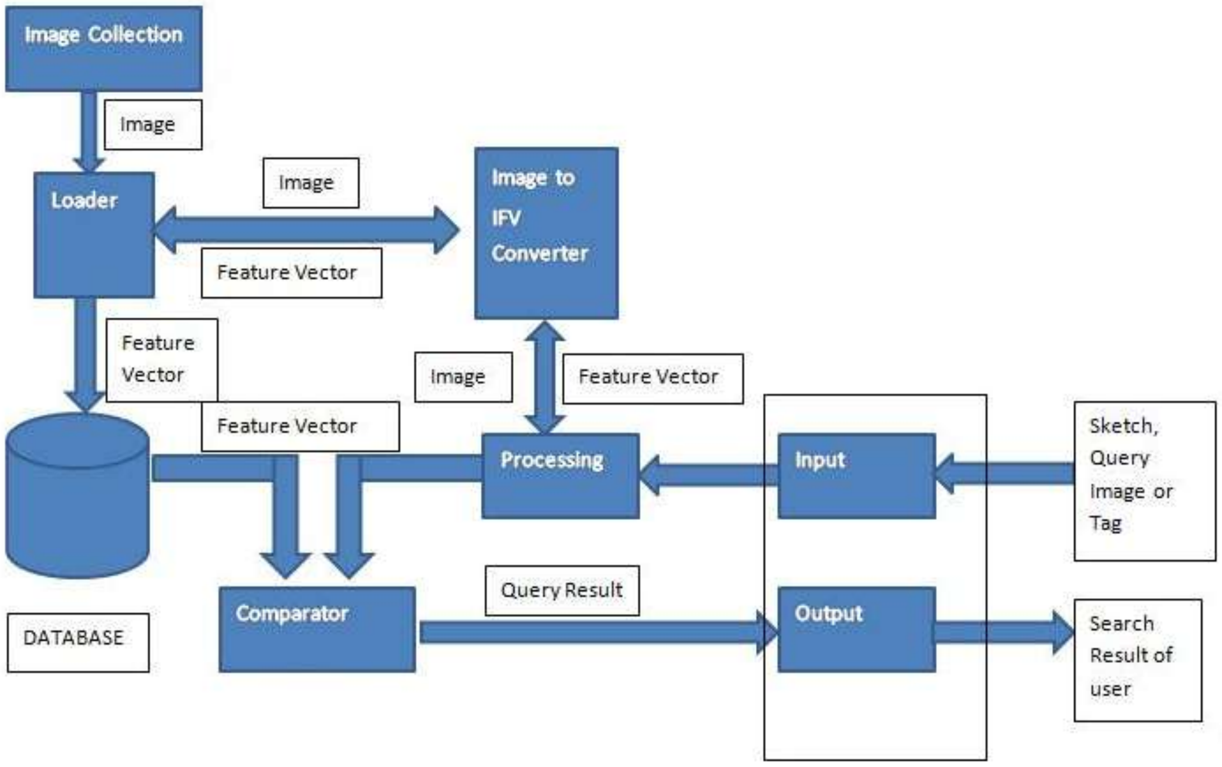


Figure 4: Architecture of FELIS

3.1.1 Loader

Loader is a small component of our system. It takes all the images from a specified path. It has a unique feature it takes images from sub folders. Initially it takes all the images from a path and gives them to Image to Feature Vector Convector. Later it stores converted images into the database. This module reads all the images of the file format *JPG, JPEG, JPE, JFIF, BMP, DIB, PNG, GIF, TIF* and *TIFF*.

3.1.2 Image to Feature Vector Converter

This is the main component of our System. This component contains various Matlab files. It converts images to their feature vectors. These feature vectors contain content of images. For color feature, it converts it into HSV histogram. For shape feature, it converts it into Distance Auto-correlogram and for texture feature it converts it into Gray-Level Co-occurrence Matrix (GLCM). Finally it returns 4 parameters to database. Following are the four parameters

i. **32 bit Hash Value:** It represents a unique key for each image. For each image it acts as a signature which represents a unique value to identify each image uniquely. This value is generated using standard MD-5 (Message Digest 5) algorithm. Using this value, image can be tracked even if it moves from one directory to another directory.

ii. **Number of Objects:** Initially image is segmented into various objects using segmentation algorithm .The total number of objects present in an image is returned by this parameter.

iii. **Color Histogram:** This represents color distribution in an image. For each image, this value is used to search images based on color pixel distribution.

iv. **Object Feature:** For each object present in the image, its color, shape and texture feature are extracted and stored into database.

3.1.3 Database

This is the repository of our system. Here all the information returned by IFVC is stored. We have used Matlab database for creation of database. In the Loading phase features of every image is stored in the database. When user queries the system, Comparator module compares query image features with all stored image features in the database. It takes very less space.

3.1.4 Comparator

Our system provides two search options.

1. Search by Image
2. Search by Object

In search by image Comparator searches the images by comparing image histogram. In image comparison minimum and maximum threshold is decided, based on threshold most relevant images are retrieved.

In object based searching first image is segmented into various objects, and then comparison is done through objects. First it compares the shape object after that compares color and texture of related object form database.

3.1.5 Graphical User Interface

This is the component which provides user with an interface through which user interacts with the system and queries the system. On GUI following options are provided.

i. **Load:** This option provides user with option to load a single file or all the images present in the folder into database. System provides user with the facility of log file generation by which log file of all the loaded images is generated containing sufficient information.

ii. **Search by Image:** This is querying interface. Once a user selects an image and clicks on search by image button, all the images with similar color distribution are retrieved and most similar results are displayed.

iii. **Search by Object:** This is another querying interface. Once user selects an image and clicks on search by object, image is segmented and all the objects inside that particular image get extracted. User can select any particular object by clicking on it and all the images containing similar objects are retrieved.

3.2 IMAGE TO FEATURE VECTOR CONVERTER [13]

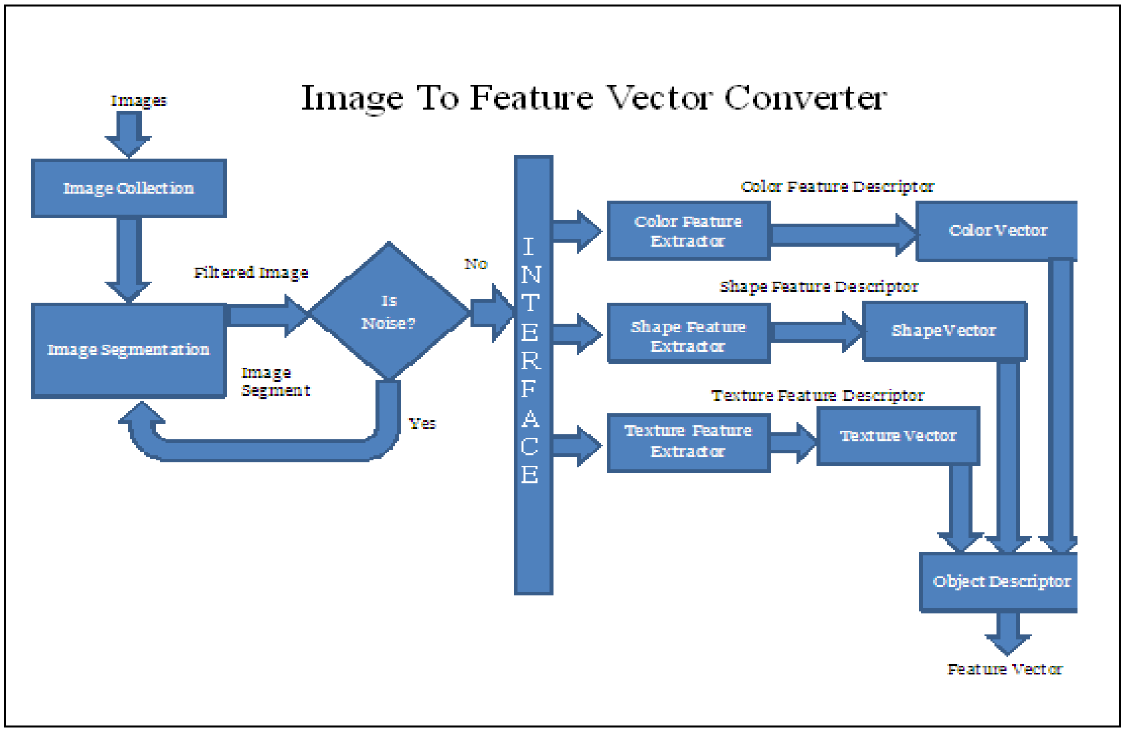


Figure 5: Image to Feature Vector Converter

Image to Feature Vector Converter is a module which receives a query image and converts it into its feature vector converter. The above module only shows object feature extractor. Following are different modules.

1. **Image Pre-processing:** It is a step which is used to remove noise content from the image. Considering basic noise content, we apply symmetric Gaussian filter to the image. Filtered image is processed further.

2. **Image Segmentation:** Object extraction is performed by doing image segmentation. We've used SRM (Statistical Region Merging Algorithm) which performs segmentation in number of iterations. Optimum results are obtained for 2^8 (= 256) iterations. After segmentation, each object is recognized as different from the background. Smaller objects are merged with larger object and neglected and considered as no objects at all.

3. **Feature Extractor:** For each object extracted, their color, shape and texture features are extracted and represented using feature vector by merging all the values. All the feature values together represent Object Descriptor.

CHAPTER 4 IMPLEMENTATION ISSUES

IMPLEMENTATION ISSUES

Following are different issues regarding design of FELIS.

1. Domain

One of serious issue is regarding choice of domain of project i.e. designing as a desktop search engine or web-based search engine. Desktop search engines are useful to organize user's personal image collection. But with growth of internet and growth in usage of photo hosting sites future of computing is cloud or network-based system. Hence it is a serious matter of choice whether the design should be web-based or desktop based.

2. Image Composition

When Image segmentation is applied to an image, all the image objects are collected separately. For e.g. if human face is segmented we get face, eyes, nose and mouth as separate objects. But we need to consider an algorithm or some way to combine these objects together to represent a face. To do this, we need to consider composition relationship. But while doing composition some of unexpected results can be obtained.

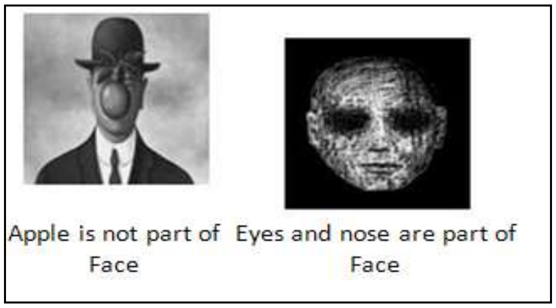


Figure 6: Challenges in Composition

For e.g. consider a picture where an apple is kept on a table and a man is standing behind it. In this case if apple and man are composed together then we will get some of unexpected result. So, whether image composition should or should not be considered is a serious issue.

3. Image Format

Since the project has been designed to work as a general purpose search engine, it needs to be able to process almost every format. Thus, which all image formats to be allowed and which all to be left is one of issue.

CHAPTER 5
SCREEN SHOTS

SCREEN SHOTS

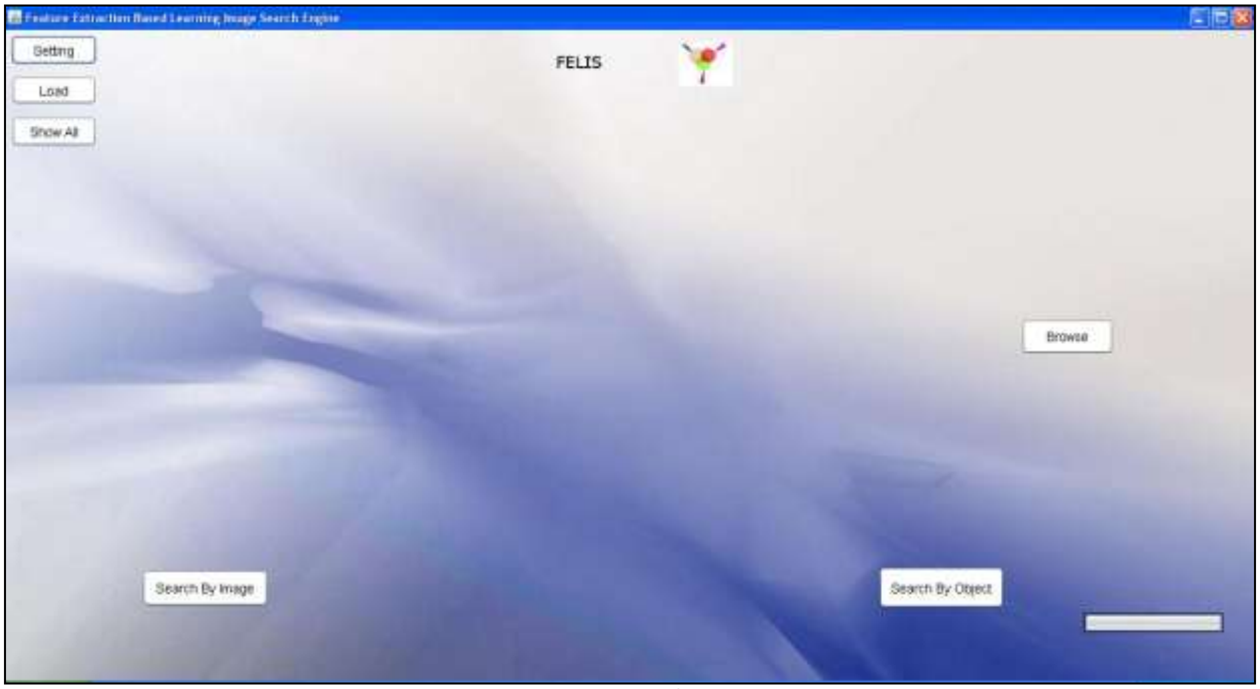


Figure 7: Homepage of the Project

Figure 7 is Homepage of our project FELIS. Logo of project contains 3 binoculars one search images based on color, second search images based on shape and third search images based on texture. Many options provided on this homepage as follows:-

Load: This option takes the image from a specified path & automatically loads that image into database.

Show All: This option show all the images present inside the database.

Setting: This option allows user to set different criteria according to his/her preference.

Browse: With this option user can browse all the images contained in various drives, folders & subfolders inside computer & can upload that particular image.

Search by Image: This option retrieves all the similar type of images to the query image from the database. This option works only based on color feature.

Search by Object: Once user selects an image and clicks on search by object, image gets segmented and all the objects inside that particular image get extracted. User can again select any particular object by clicking on it and all the images containing similar objects are retrieved.

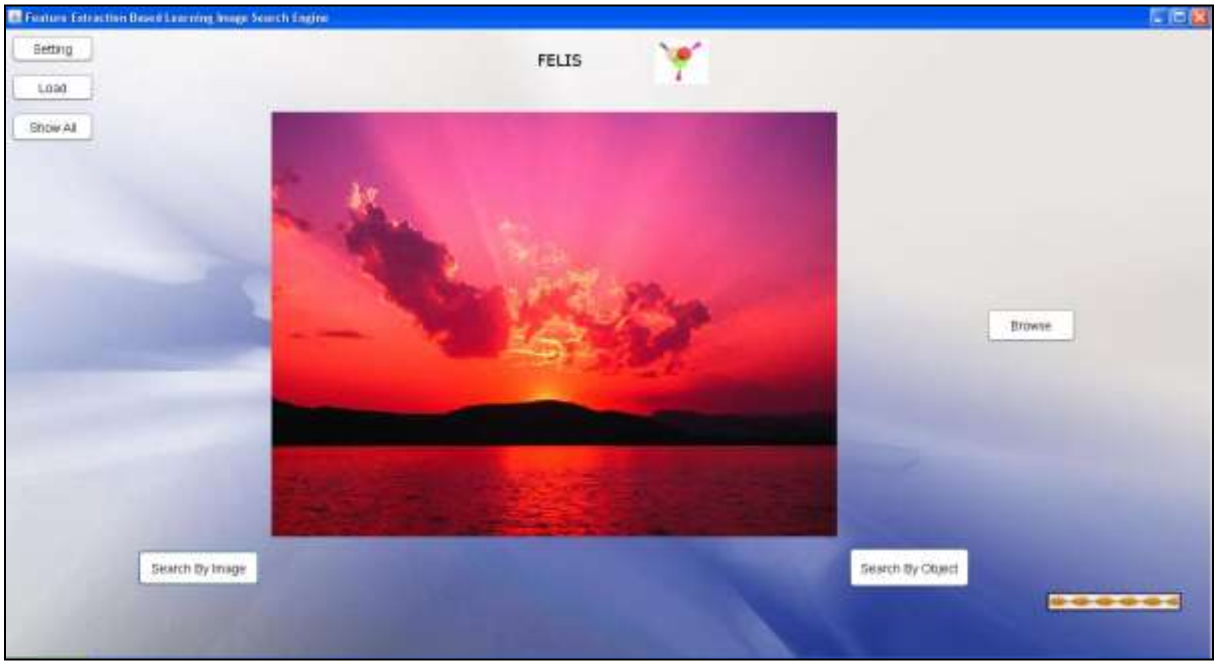
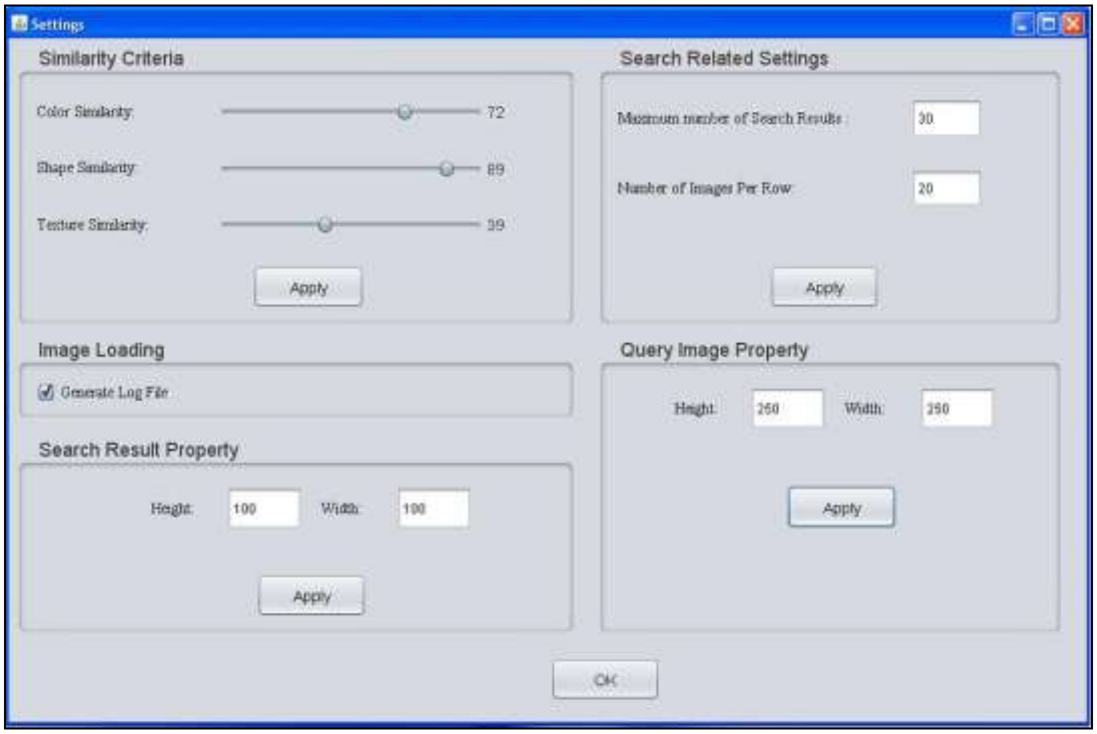


Figure 8: Homepage during Searching

In figure 8, when user clicks on the Browse button & selects any image for uploading from different drives then that image gets uploaded to the center part of Homepage.



Settings

Similarity Criteria

Color Similarity: 72

Shape Similarity: 89

Texture Similarity: 39

Apply

Search Related Settings

Maximum number of Search Results : 30

Number of Images Per Row: 20

Apply

Image Loading

☑ Generate Log File

Search Result Property

Height: 100 Width: 100

Apply

Query Image Property

Height: 250 Width: 250

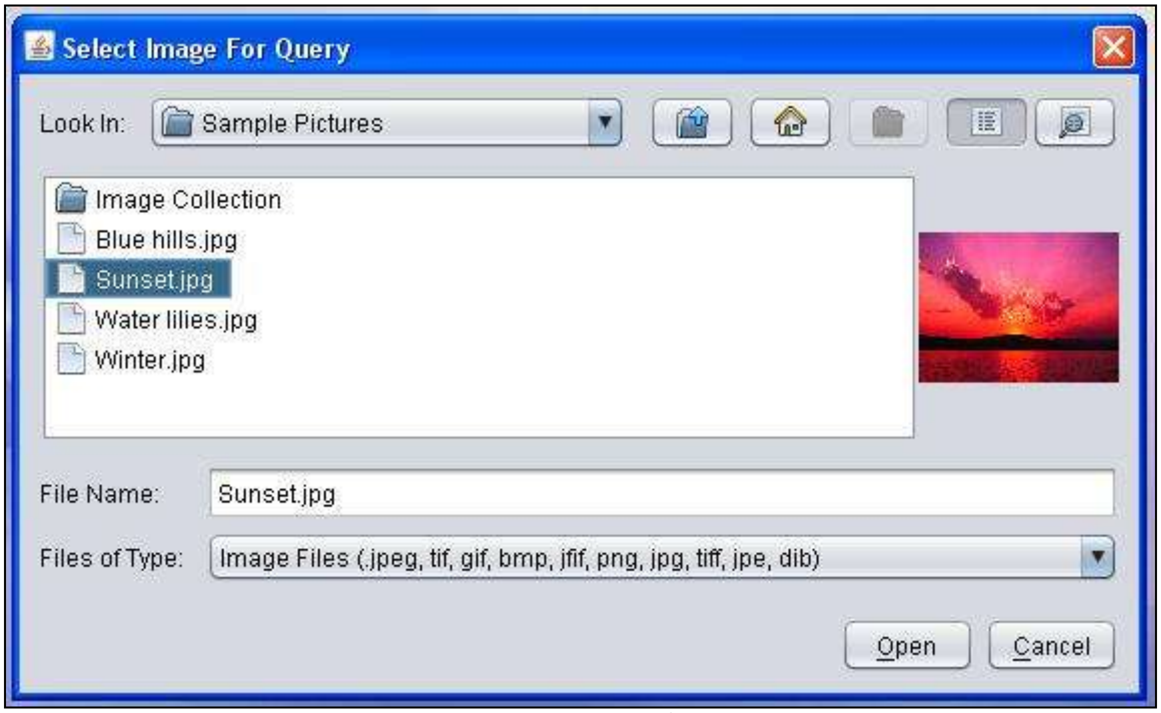
Apply

OK

Figure 9: Setting Window

Figure 9 i.e. Setting Option provides various setting options to the user i.e. In Similarity Criteria sub-option it provides similarity criteria for color feature, shape feature & texture feature with which user can vary the values of any feature.

In Image Loading sub-option it generates log files of all the loaded images which contain needed information about that image.
In Search Result Property sub-option it provides Height & Width of image through which user can set size of search resulting images.
In Search Results Settings sub-option user can set maximum number of search results page & number of images per row as per his requirement.
In Query Image Property sub-option user can set height & width of query image.



File Name: Sunset.jpg

Files of Type: Image Files (.jpeg, tif, gif, bmp, jfif, png, jpg, tiff, jpe, dib)

Figure 10: File chooser window for searching

In figure 10, User can search any image from any drives or any folder in computer. When he will select particular image then its preview will be shown. Here user selects Sunset.jpg then corresponding image preview get display.

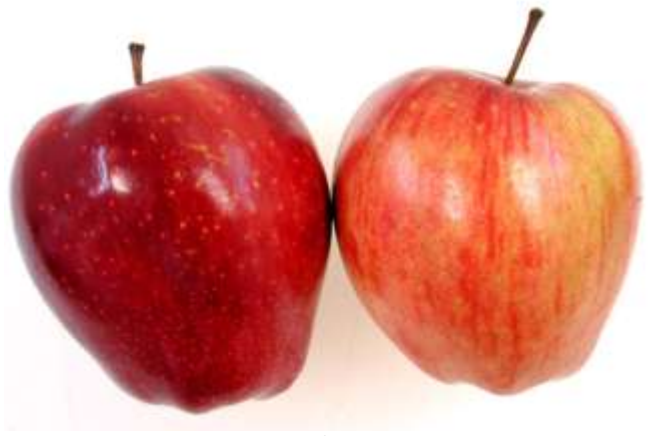


Figure 11: Image for search by Object

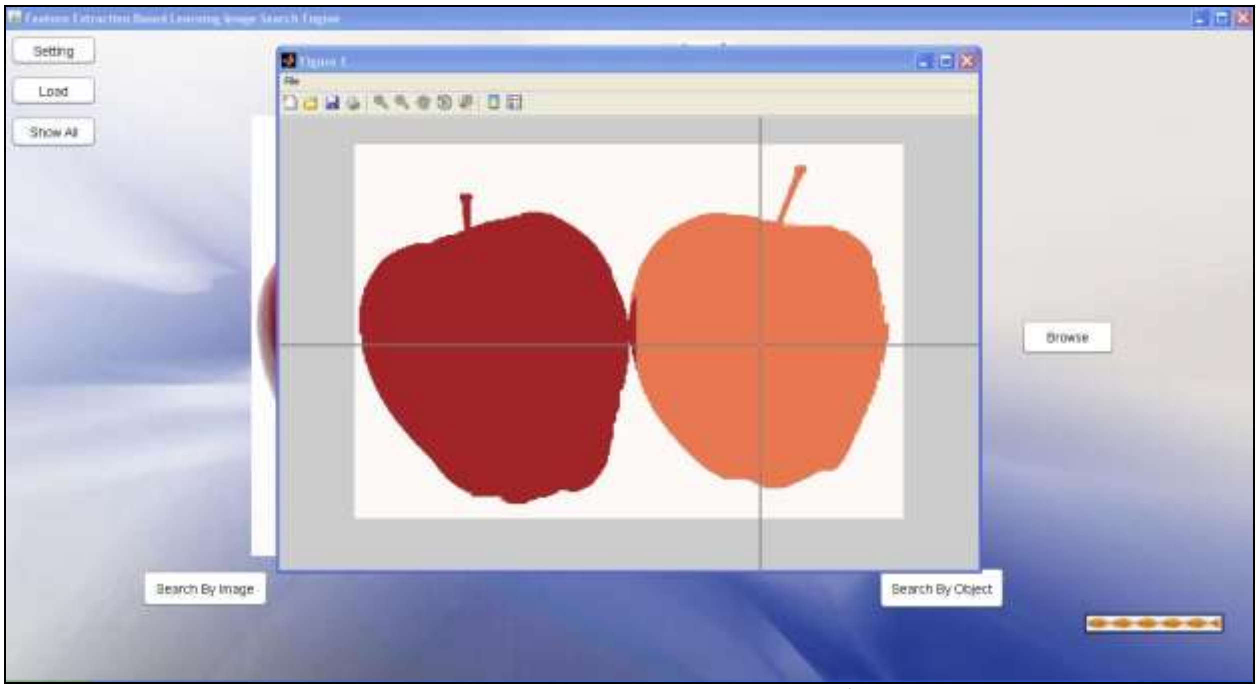


Figure 12: Search by Object: Object Selection Interface

In figure 12, when user selects any image & clicks on search by object then it will show various objects present in the image. As in above image objects of 2 apples get displayed. Then user can select anywhere in that image. Original image is shown in figure 11.

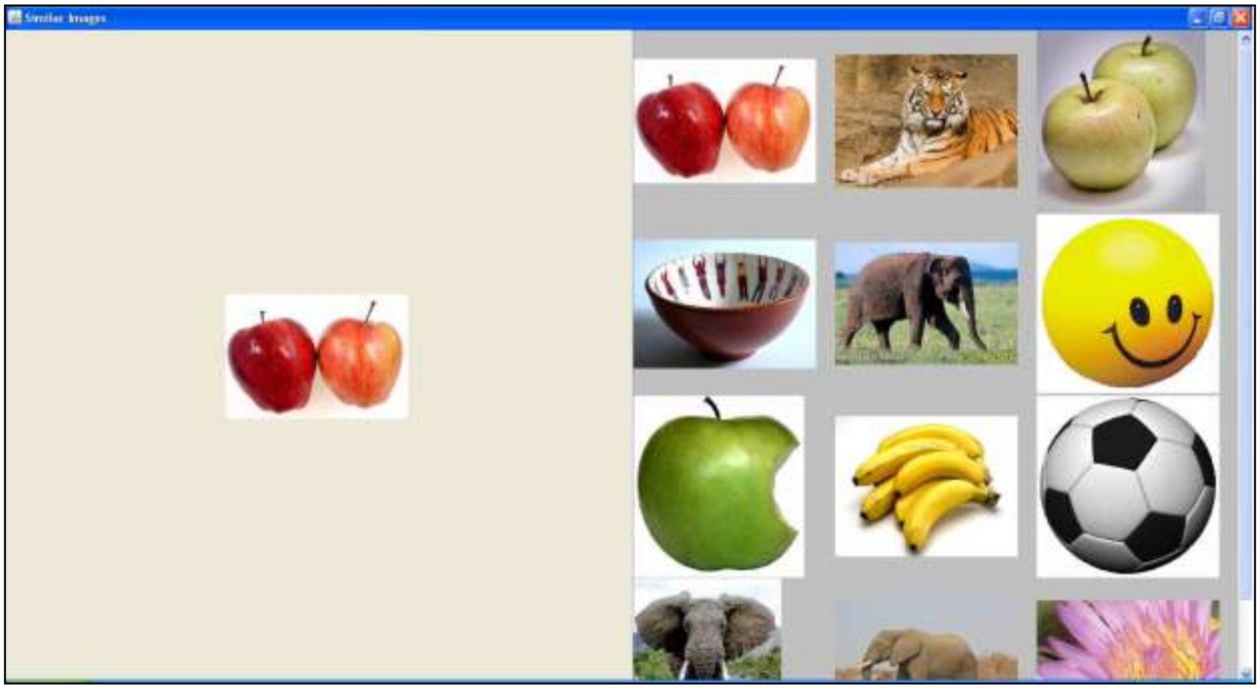


Figure 13: Search by Object Result

In figure 13, related images like query image containing that particular type of object by color, texture and shape get displayed.

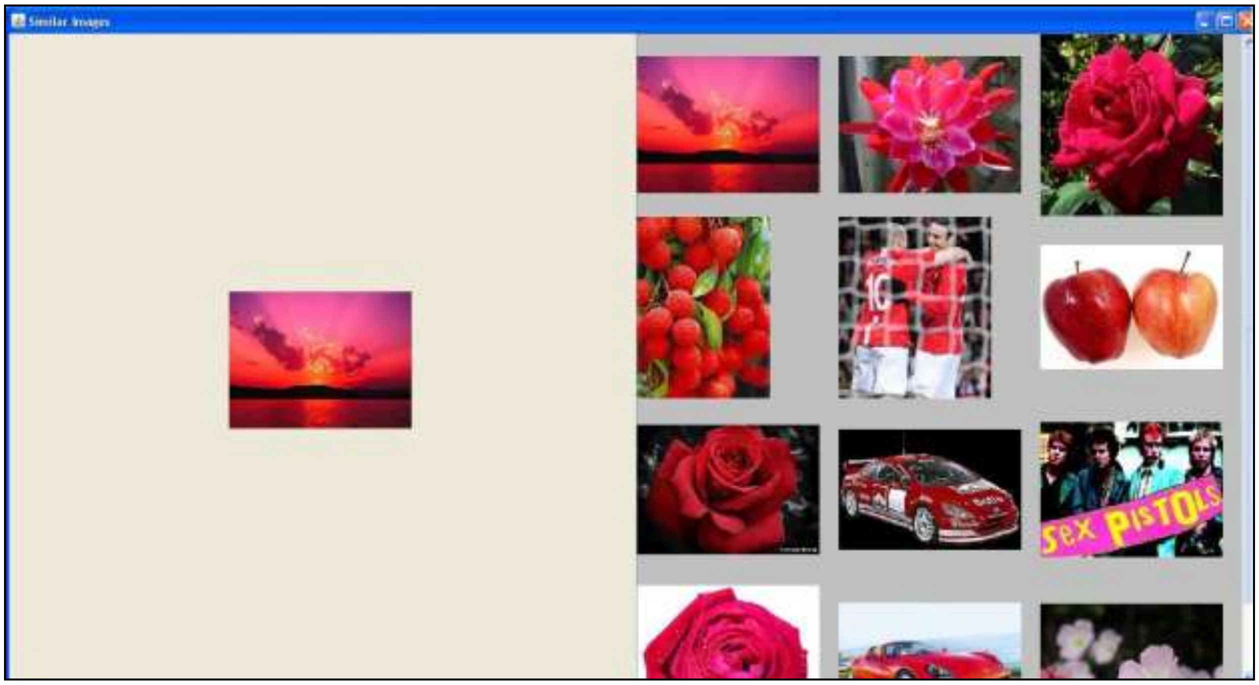
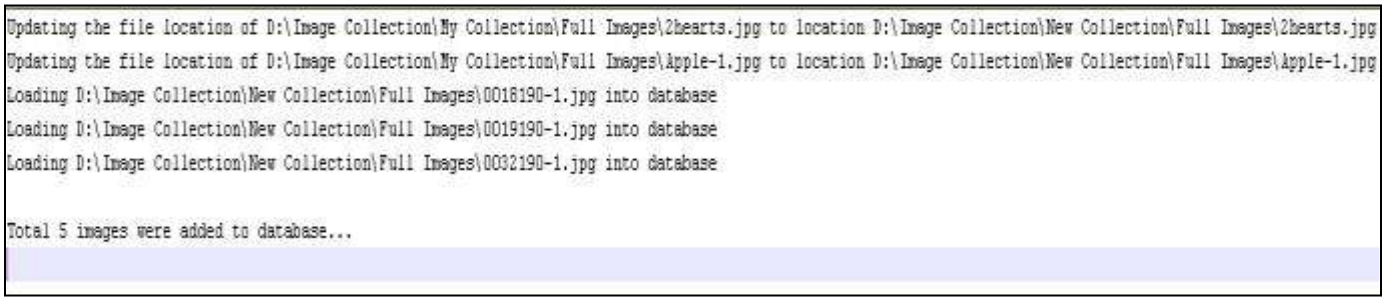


Figure 14: Search by Image Result

In figure 14, images related to the query image will get retrieved by the database based on color feature.



```
Updating the file location of D:\Image Collection\My Collection\Full Images\2hearts.jpg to location D:\Image Collection\New Collection\Full Images\2hearts.jpg
Updating the file location of D:\Image Collection\My Collection\Full Images\Apple-1.jpg to location D:\Image Collection\New Collection\Full Images\Apple-1.jpg
Loading D:\Image Collection\New Collection\Full Images\0018190-1.jpg into database
Loading D:\Image Collection\New Collection\Full Images\0019190-1.jpg into database
Loading D:\Image Collection\New Collection\Full Images\0032190-1.jpg into database

Total 5 images were added to database...
```

Figure 15: Log File Generated

In figure 15, log file related to loaded images gets generated. Each log file contains related information about particular image.

CHAPTER 6
TESTING

6.1 INTRODUCTION TO TESTING

Software testing is a critical phase of software development cycle which is generally performed along with development process. Testing is performed to check whether the software is up to expected quality mark or not. Testing involves testing each and every aspect of the software including specification, design, code generation and code maintenance.

Following are the objectives of testing.

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as yet undiscovered error.
- Successful test is one that uncovers an as yet undiscovered error.

In our project testing, we are executing the program with the intent of finding an error or some unexpected behavior. During the process of testing, we are applying different sets of input to the each module of the program with the intention to find and study behavior of the program and to check whether the output generated is independent of sequence of the input or not. Following are some of test cases executed in the program.

6.2 TEST CASES

6.2.1 Testing on Image Segmentation

Image segmentation algorithm was supposed to extract different objects from the image. For very smaller objects the segmentation was supposed to ignore their presence and treat them as a part of background image only.

Case 1: In figure 16 given below, object heart is very distinct from the background and is quite large in size. Hence upon segmentation, image is divided into 2 parts one representing heart & other representing background of the image.

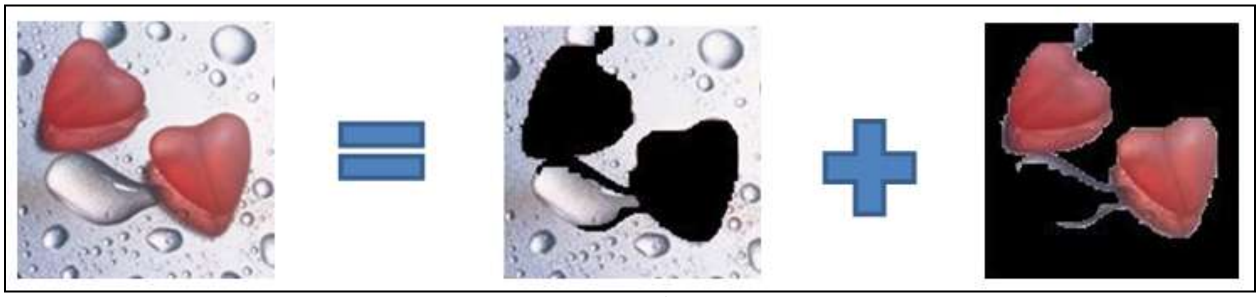


Figure 16: Perfect Segmentation

Case 2: In figure 17, object is distinct from the background. But its size is very small compared to size of entire image and it is completely covered by background part from all sides. Hence it is completely ignored during segmentation and only one objects i.e. background is extracted.

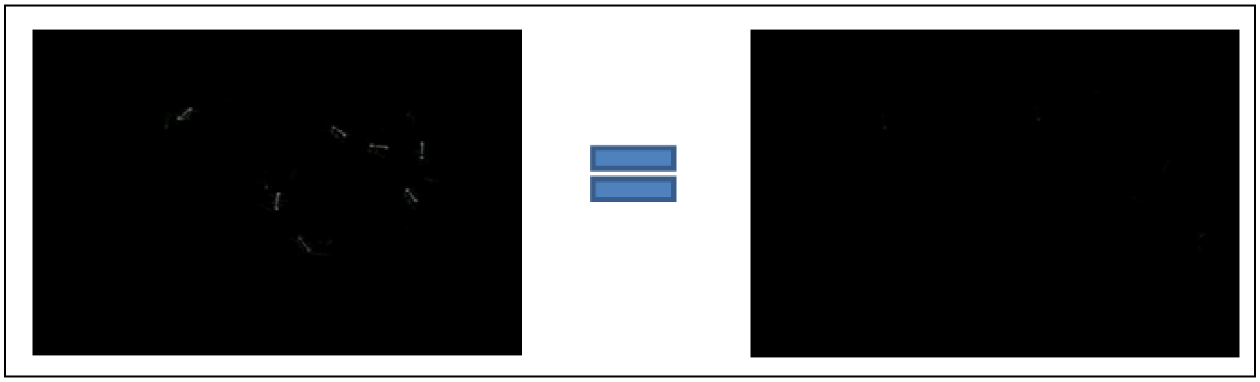


Figure 15: Segmentation and small Object Removal

6.2.2 Testing on User Interface

- User interface was tested for different users and their opinion was taken regarding navigation of the system and ease of use of the system. It was found that almost all the users were satisfied with the design of the system and found it very simple to learn.

- Testing on creation of database was done in which images were loaded after deleting the database file. It was found that system creates a new database file and records the entry of only those images which were loaded into system.

- Testing was done on show all images option in which database file does not contain any image and it was found that the output screen contains a blank window and no exceptions occur.

- Testing was done on loading an image into database and it was found that system works properly when a single file or an entire folder is selected or if no folder is selected at all.

- Testing was done on log file generation and it was found that log files are generated only after checkbox is selected otherwise existing log file remains as it is. However if the log file is already present and images are loaded again the log file is overwritten.

- Testing was done to study the behavior of the system in the case of new input when previous input is already being processed. It has been observed that next input is processed only if previous input is already processed.

- Testing was done on File selector and it was found that for the entire allowed image file formats, their preview is displayed properly.

- Testing was done in which no image was selected for querying. In that case, query image remains the new query image.

CHAPTER 7

LIMITATIONS AND FUTURE SCOPE

7.1 LIMITATION

As stated earlier, major success of the system depends on the accuracy with which segmentation is performed which is prone to illumination effect. Thus, segmentation results are largely affected by the illumination effect. Due to illumination, one single object is treated as number of objects. In worse cases, no of segments present are so large (> 10000) that system crashes. Feature representation algorithm were implemented and tested for different scenario's but feature comparison algorithms except for color representation do not yield very accurate results.

Also, for desktop users, initial loading of images takes large amount of time.

7.2 FUTURE SCOPE

Based on our study work and limitation of the system, we came to conclusion that the system can be extended in different areas to improve it to very large extent.

1. **Auto-tagging:** It is difficult for user to specify the feature every time he wants to search. Hence, auto-tagging feature can be incorporated. Thus, once an object can be tagged it can later be searched by tag name itself.

2. **Web Based System:** Desktop systems will soon be replaced by cloud system due to widespread of internet. And hence, system can be extended for cloud system. On internet everyday a very large amount of multimedia data is generated which can be fed into system and tagging can be added to create a very big knowledge base system which can be used anywhere including tracking cars with specific number or tracking a person.

3. **Automatic Image Loading using crawler:** Apart from this, a crawler can be designed if it is to be developed as a web based search engine so that it can automatically load the images without user specifying them.

CHAPTER 8
CONCLUSION

CONCLUSION

The intended system i.e. Feature Extraction based Learning image search engine was implemented successfully for desktop systems with the expected functionalities.

Although the system does not produce 100% accurate results but still produces results which are much better than existing search engine. Object based searching provides a new look of images by content by providing content as object itself rather than a specific feature and hence providing a notion of more accurate search engine.

The system can be distributed among different types of users and can be tested to work and their feedback can be collected to improve the system functionality. It’s just first step towards developing an intelligent searching system which can be further extended to develop an entirely user preference specific intelligent system.

CHAPTER 9
REFERENCES

REFERENCE

[1].*Features for Image Retrieval'*, Thomas Deselaers, Thesis

[2].'*Blobworld: Image Segmentation Using Expectation-Maximization and Its Application to Image Querying.*' Chad Carson, Serge Belongie, Hayit Greenspan, and Jitendra Malik. IEEE Transactions on Pattern Analysis and Machine Intelligence, VOL. 24, NO. 8.

[3].'*PATSEEK: Content Based Image Retrieval System for Patent Database'*, AvinashTiwari, VeenaBansal, Industrial and Management Engineering Department, IIT Kanpur

[4].'Photobook: Content Based Manipulation of Image Database', A. Pentland, R. W. Picard, S. Sclaroff, MIT, Perceptual Computing Section, Report No 255, November 1993

[5].'Efficient and Effective Querying by Image Content', C. Faloutsos, W. Equitz, M. Flickner, W. Niblack, D. Petkovic, R. Barber, IBM Research Division, Almaden Research Center.

[6].'SIMBA: Search by Image Appearance', S. Segglekow, M. Schael and H. Burkhardt, Institute of Pattern Recognition & Image Processing, Albert Ludwigs University, Germany.

[7].'*A Survey On: Content Based Image Retrieval Systems'*, NidhiSinghai and Prof. Shishir K. Shandilya, Department of CSE NRI Institute of Science & Technology, International Journal of Computer Applications (0975 – 8887) Volume 4 – No.2, July 2010

[8].'*CIRES: A System for Content-based Image Retrieval in Digital Image Libraries*', QasimIqbal and J. K. Aggarwal, Computer & Vision Center, The University of Texas at Austin, Texas.

[9]. '*Statistical Region Merging*', Richard Nock & Frank Nelson, IEEE Transaction on Pattern Recognition & Machine Intelligence, Volume 26, Number 11, November 2004.

[10]. '*Color Models*', Michael Swain, Dana Ballard, International Journal of Computer Vision, 7:1, 11-32(1991).

[11]. 'Textural Features Corresponding to Visual Perception', Amura, Hideyuki Mori, ShunjiYamawaki, Takashi.

[12]. '*A Novel Shape Representation and Retrieval Algorithm: Distance Autocorrelogram*', Jie-xianZeng, Yong-gang Zhao, Xiang Fu, Nanchang Hang Kong University, China, Journal of Software, volume 5, number 9.

[13]. '*Design & Architecture of FELIS*', Ms. Sukhwant Kaur, Ms. Sandhya Pati, Mr Mahesh Gupta, Ms Sneha Kamat, Mr Abdul Mohsin, International Conference on Emerging Trends in Computer Science & Information Technology, February 2011, GGI, India

[14]. http://en.wikipedia.org/wiki/MD5

[15]. http://aa-lab.cs.uu.nl/cbirsurvey/cbir-survey/node30.html

[16]. http://aa-lab.cs.uu.nl/cbirsurvey/cbir-survey/node34.html

[17]. http://aa-lab.cs.uu.nl/cbirsurvey/cbir-survey/node36.html