**ABSTRACT**

Image processing is a form of signal processing in which it takes image as input and produces the output which may be in the form of images or the set of characteristics or parameters related to the image. Content based image retrieval is an automatically extraction of the images based on the color, texture and shape. Image retrieval is concerned with the accessing of the desired images from large and varied image collections.Query By Image Content (QBIC) is the system that can filter images based on their content. The term 'content' in this context might refer to colors, shapes, textures, or any other information that can be derived from the image itself. QBIC 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. One of the key issues with any kind of image processing is the need to extract useful information from the raw data before any kind of reasoning about the image’s contents is possible. There is a growing interest in QBIC because of the limitations inherent in metadata-based systems, as well as the large range of possible uses for efficient image retrieval. QBIC system is able to serve queries ranging from scenes of purely natural objects such as vegetation, trees, sky, etc. to images containing conspicuous structural objects such as buildings, towers, bridges, etc. Different implementations of QBIC make use of different types of user queries.

**Acknowledgement**

We have great pleasure in presenting the report on “**Query By Image Content using Content Based Image Retrieval**”. We take this opportunity to express my sincere thanks towards the guide [Prof. Prajakta Bhangale] , C.R.C.E, Bandra(W) , Mumbai, for providing the technical guidelines, and the suggestions regarding the line of this work. We enjoyed discussing the work progress with her during our visits to department.

We thank Prof. Mahesh Sharma, Head of Information Technology Dept., Principal and the management of C.R.C.E., Mumbai for encouragement and providing necessary infrastructure for pursuing the project.

We also thank all non-teaching staff for their valuable support, to complete our project.

**Contents**

List of Figures……………………………………………………………………...06

List of Tables……………………………………………………………………….07

1. Introduction………………………………………………………………...09
2. Literature Review…………………………………………………………..12
   1. Feature Extraction…………………………………………………….13

2.1.2 Haar Wavelet Transform……………………………………………13

* 1. Clustering……………………………………………………………16
  2. Euclidean Distance..………………………………………………….19

1. Problem statement……………………....………………………………….20
2. Project Description………………………....………………………………23
   1. Overview of the project……………………………………………....24
   2. Module Description………………….………………………………..25
      1. Use Case Diagram………………………….…………….…….30
      2. Data Flow Diagram…………………….……………………....32
      3. Activity Diagram………………………….……………………33
3. Testing Technology……………………………….………………………..36
   1. Unit Testing……………………………….……….………………....37
   2. Performance Testing………….………….…………………………...38
   3. Load/Stressing Testing…………………….………………………….39
   4. Acceptance Testing……………………….…………………………..39
4. Implementation Details……………………….……………………………39
   1. Methodology…………………………….…....………………...……40
      1. Haar Wavelet transform…………….………...……………….41
      2. Progressive Retrieval Strategy……………….......…………...51
      3. Clustering of images………….......…………...……………...52
5. System requirements………………………………….……......………….54
   1. Hardware requirements……………………......….………………....55
   2. Software requirements…………………...........……...………...…..55
6. Conclusion & Future Scope………………………………...…...………..57

9. Bibliography ……………………………….…….………..………...……59

**List Of Figures**

|  |  |  |
| --- | --- | --- |
| **Fig No.** | **Fig Name** | **Page Number** |
| Fig 1 | Haar Wavelet | 1 |
| Fig 2 | Original Image(haar) | 14 |
| Fig 3 | Haar Coefficients | 14 |
| Fig 4 | Fewer Haar Coeffficients | 15 |
| Fig 5 | Reconstructed Image | `5 |
| Fig 6 | Haar Transform Coefficient Representation1 | 15 |
| Fig 7 | Haar Transform Coefficient Representation2 | 16 |
| Fig 8 | Haar Transform Coefficient Representation3 | 16 |
| Fig 9 | Cluster Plot | 17 |
| Fig 10 | Flow Chart | 25 |
| Fig 11 | Indexing Input | 26 |
| Fig 12 | Start Indexing | 27 |
| Fig 13 | Calculated wavelet | 28 |
| Fig 14 | Clustered Images | 29 |
| Fig 15 | User input To search Image | 30 |
| Fig 16 | Output | 31 |
| Fig 17 | Use Case Diagram | 32 |
| Fig 18 | DFD0 | 33 |
| Fig 19 | DFD1 | 33 |
| Fig 20 | DFD1.1 | 34 |
| Fig 21 | Activity Diagram | 35 |
| Fig 22 | Block Diagram of Content Based Image Retrieval | 43 |
| Fig 23 | Haar wavelet Transform | 44 |
| Fig24 | Haar Forward Transform For 8 Element signal | 45 |

**List Of Tables**

|  |  |  |
| --- | --- | --- |
| **Table No 1** | **Name of Table** | **Page No** |
| 6.1 | Selecting and matching the images with indexed database | 41 |

**Chapter1: Introduction**

**Introduction**

Earlier, we were using a technique of assigning a keyword to the similar types of images and retrieving that images manually by using a keyword assigned to the images. But this technique was time consuming and the output was both inefficient and ineffective. To remove this drawback and to improve the performance of the system QBIC technique has been proposed. The color feature is widely used in the QBIC system. Thus content-based image retrieval system is a piece of software that implements QBIC.

Content-Based Image Retrieval (QBIC) is considered as the process of retrieving desired images from huge databases based on extracted features from the image themselves (without resorting to a key word). Features are derived directly from the images and they are extracted and analyzed by means of computer processing.QBIC is a bottleneck of the access of multimedia databases that deal with text, audio, video and image data which could provide us with enormous amount of information . Many commercial and research QBIC systems have been built and developed. Content based image retrieval, allowing to automatically extract targets according to objective visual contents of image itself(e.g. color, texture and shape) has become increasingly attractive, in Multimedia Information Service System *(MISS).* With appealing time frequency localization and multi-scale properties, wavelet transform proved to be effective in visual feature extraction and representation. It can be used to characterize textures using statistical properties of the gray levels of the points/pixels comprising a surface image.

In QBIC, wavelet approaches mainly include wavelet histogram and wavelet moment of image, etc.. Wavelet transform can be used to characterize textures using statistical properties of the gray levels of the pixels comprising a surface image. The wavelet transform is a tool that cuts up data or functions or operators into different frequency components and then studies each component with a resolution matched to its scale.

We used D4 and Haar wavelet transforms to decompose color images into multilevel scale and wavelet coefficients, with which we perform image feature extraction and similarity match by means of F-norm theory. We also present a progressive retrieval strategy, which contributes to flexible compromise between the retrieval speed and the recall rate. The retrieval performances are compared with the existing wavelet histogram technique. The efficiency in terms Recall rate and retrieval speed is tested with five types of images and the results reflect the importance of wavelets in QBIC.Fnorm theory along with progressive retrieval strategy improves retrieval performance.

**Motivation**

Image Retrieval technique is used almost all big Image Processing areas like Medical applications, Military & Remote Sensing System, Image Search Engines as well as in all the smartphones and mobile world. Many systems have been developed for Image Retrieval. The system uses content of the image to retrieve Image from database. Here is this system we are going to take Color, Shape and texture of the Image as the content of the image instead of visual appearance of the image This thing motivated us to choose this topic. Then after we came to know that almost all the systems going to deal with features like color and shape or shape and texture or color and texture so we are going to develop a system which works on three features(color texture and shape) of the image.

* 1. **Objectives**
* To study and compare the various algorithms for Feature Extractions from the image.
* Improve Database Management using proper clustering algorithm

**Chapter 2 : Literature Review**

**Chapter 2**

**Literature Review**

Numerous works are available in the literature related with the Content-Based Image Retrieval. Several reviews of the literature on image retrieval have been published, from a variety of different viewpoints. A brief summary of some of the significant researches is presented below:

Enser [1995] reviews methods for providing subject access to pictorial data, developing a four-category framework to classify different approaches. He discusses the strengths and limitations both of conventional methods based on linguistic cues for both indexing and search, and experimental systems using visual cues for one or both of these. Aigrain et al [1996] discuss the main principles of automatic image similarity matching for database retrieval, emphasizing the difficulty of expressing this in terms of automatically generated features. They review a selection of current techniques for both still image retrieval and video data management. Eakins [1996] proposes a framework for image retrieval, classifying image queries into a series of levels, and discussing the extent to which advances in technology are likely to meet users’ needs at each level. Idris and Panchanathan [1997] provide an in-depth review of CBIR technology, explaining the principles behind techniques for colour, texture, shape and spatial indexing and retrieval in some detail. De Marsicoi et al [1997] also review current CBIR technology, providing a useful feature-by-feature comparison of 20 experimental and commercial systems.

A number of structures has been proposed for handling multi-dimensional point data. Antoine Guttman was one of the first persons to propose them. In 1984, Guttman published a book[Gutt84] in which he presented a data structure called R-Tree (Rectangle Tree) that represents data objects by intervals in several dimensions.

**Present Investigation**

Image Query is a Content Based Image Retrieval System using Wavelet Transform and K-Means Clustering developed in Java

. "Content-based" means that the search will analyse the actual contents of the image. The term 'content' in this context might refer colors, shapes, textures, or any other information that can be derived from the image itself.

**2.1 Haar Wavelet Transform:**

In mathematics, the **Haar wavelet** is a sequence of rescaled "square-shaped" functions which together form a [wavelet](http://en.wikipedia.org/wiki/Wavelet) family or basis. Wavelet analysis is similar to [Fourier analysis](http://en.wikipedia.org/wiki/Fourier_analysis) in that it allows a target function over an interval to be represented in terms of an[orthonormal](http://en.wikipedia.org/wiki/Orthonormal) function basis. The Haar sequence is now recognised as the first known wavelet basis and extensively used as a teaching example.

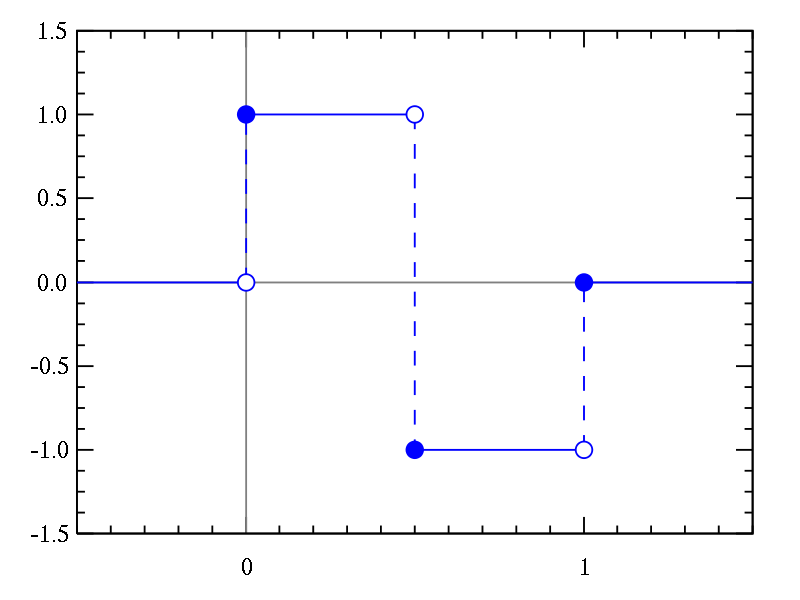


Fig1: Haar Wavelet

**Working of Haar:**

The easiest of all discrete wavelet transformations is the Discrete Haar Wavelet Transformation (HWT). Let's motivate it's construction with the following example:

Suppose you had the eight numbers 100, 200, 44, 50, 20, 20, 4, 2 (these could be grays cale intensities) and you wanted to send an approximation of the list to a friend. Due to bandwidth constraints (this is a really old system!), you are only allowed to send your friend four values. What four values would you send your friend that might represent an approximation of the eight given values?

There are obviously many possible answers to this question, but one of the most common solutions is to take the eight numbers, two at a time, and average them. This computation would produce the four values 150, 47, 20, and 3. This list would represent an approximation to the original eight values.

Unfortunately, if your friend receives the four values 150, 47, 20, and 3, she has no chance of producing the original eight values from them - more information is needed. Suppose you are allowed to send an additional four values to your friend. With these values and the first four values, she should be able to reconstruct your original list of eight values. What values would you send her?

Suppose we sent our friend the values 50, 3, 0, and -1. How did we arrive at these values? They are simply the directed distances from the pairwise average to the second number in each pair: 150 + 50 = 200, 47 + 3 = 50, 20 + 0 = 20, and 3 + (-1) = 2. Note that if we subtract the values in this list from the pairwise averages, we arrive at the first number in each pair: 150 - 50 = 100, 47 - 3 = 44, 20 - 0 = 20, and 3 - (-1) = 4. So with the lists (150,47,20,3) and (50,3,0,-1), we can completely reconstruct the original list (100,200,44,50,20,20,4,2).

**Image Compression:**The Haar transform can be used in lossy image compression:

|  |  |  |
| --- | --- | --- |
| https://unix4lyfe.org/haar/cat-original.png | → | https://unix4lyfe.org/haar/cat-coeff.png |
| Fig:2. Original image(haar) |  | Fig3: Haar coefficients |
|  | | |
| https://unix4lyfe.org/haar/cat-coeff-5pct.png | → | https://unix4lyfe.org/haar/cat-output.png |
| Fig4: Fewer Haar coefficients |  | Fig5: Reconstructed image |

More examples of images are given below:

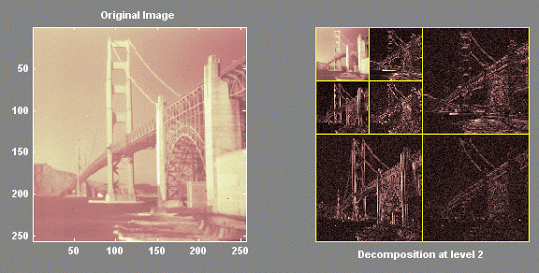
****

Fig6: Haar transform coefficient Representation1

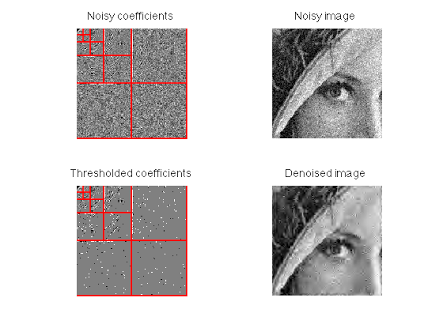


Fig 7: Haar transform coefficient Representation2

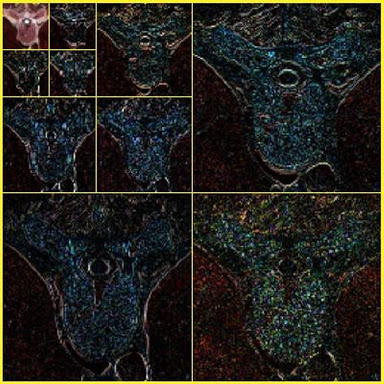


Fig 8: Haar transform coefficient Representation3

**2.2 Clustering:**

**2.2.1 Cluster:**

One important method for data compression and classiﬁcation is to organize data points in clusters: A cluster is a subset of the set of data points that are close together, using some distance measure. • A loose deﬁnition of clustering could be the process of organizing data into groups whose members are similar in some way. A cluster is therefore a collection of data points which are “similar” between them and are “dissimilar” to the data points belonging to other clusters • One can compute the mean value of each cluster separately, and use the means as representatives of the clusters. Equivalently, the means can be used as basis vectors, and all the data points are represented by their coordinates with respect to this basis. • Clustering algorithms can be applied in many ﬁelds: – Marketing: ﬁnding groups of customers with similar behaviour given a large database of customer data containing their properties and past buying records; – Biology: classiﬁcation of plants and animals given their features; – WWW: document clustering; clustering weblog data to discover groups of similar access patterns. • There are several methods for computing a clustering. One of the most important is the k-means algorithm.

**2.2.2 Clustering:**

Clustering is the process of partitioning a group of data points into a small number of clusters. For instance, the items in a supermarket are clustered in categories (butter, cheese and milk are grouped in dairy products). Of course this is a qualitative kind of partitioning. A quantitative approach would be to measure certain features of the products, say percentage of milk and others, and products with high percentage of milk would be grouped together.

k-means clustering is a method of classifying/grouping items into k groups (where k is the number of pre-chosen groups). The grouping is done by minimizing the sum of squared distances (Euclidean distances) between items and the corresponding centroid.

A centroid is "the centre of mass of a geometric object of uniform density", though here, we'll consider mean vectors as centroids.

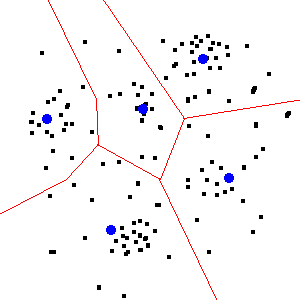


Figure 9: Cluster Plot

Figure 9. A clustered scatter plot. The black dots are data points. The red lines illustrate the partitions created by the k-means algorithm. The blue dots represent the centroids which define the partitions.

**2.2.3 The Reason behind choosing K-Means for Clustering algorithm:**

The reason we are using only k-means algorithm for clustering and no other algorithm Because of its simplicity i.e. k-means is very easy to implement and the best advantage of k-means is that it works very well when huge data sets are involved.

Further saying that it does not work very well when less images are available if less images are present then we can find them the old fashioned way i.e. by just looking at them no software is required for little images hence k-means plays a good role of clustering in our software.

**2.2.4 Use of Euclidian Distance Formula in K-Means algorithm:**

K-Means procedure - which is a vector quantization method often used as a clustering method - does not explicitly use pairwise distances at all (in contrast to hierarchical and some other clustering which allow for arbitrary proximity measure). It amounts to repeatedly assigning points to the closest centroid thereby using Euclideandistance from centroid. However, K-Means is implicitly based on pairwise Euclideandistances between points, because **the sum of squared deviations from centroid is equal to the sum of pairwise squared Euclidean distances divided by the number of points**. The term "centroid" is itself from Euclidean geometry. It is multivariate mean in euclidean space. Euclidean space is about euclidean distances. Non-Euclidean distances will generally not span Euclidean space. That's why K-Means is for Euclidean distances only.

**2.3 Euclidian Distance formula for calculating the the distance between two images:**

It can be used for the any dimensional measure or to calculate the distance between the two entities.

In one dimension, the distance between two points on the real line is the absolute value of their numerical difference. As shown below:



For two dimensional:

In the Euclidean plane if p(p1,p2) q(q1,q2) then the distance between them can be given as



**Chapter 3: Problem Statement**

**Problem Statement**

Content based image retrieval is an automatically extraction of the images based on the color, texture and shape. There are three main techniques of image retrieval, which are **color based,** texture based and shape based. Color based technique is again divided into four types which are conventional color histogram, fuzzy color histogram, color correlogram and color shape based technique. Texture based technique is again divided into four types which are steerable pyramid, contour let transform, Gabor wavelet transform and complex directional filter bank. There are several techniques of shape based image retrieval which are chain code, polygonal approximation, curvature, Fourier descriptor, radii method and moment descriptor.

**3.1 Drawbacks of the previous system :-**

Following are the problems that occur in previous methods who used two factors(shape and color or shape and text) as the content of the image[8].

* If two factors are used the another factors are not used
* All the factors are not considered
* There will not be availability of all the features to extract

**3.2 Existing System:**

The existing system to use two factors to extract the images its not enough to extract the sufficient coefficients from the image which can define the entire image for the purpose of our task. Which leads to improper outcome at the end.

**Disadvantages:**

1. The systems cannot extract all necessary features.

2. proper outcome will not possible in third factor comes in play.

**3.3 Possible Solution to the above problem**

* In our project, we are going to consider three factors to extract the features from the image.
* The use of these will take out the confusion while extracting comparing images when third factor comes in play.

**Chapter 4: Project Description**

**PROJECT DESCRIPTION**

**4.1 Overview of The Project:**

Design involves identification of classes, their relationships as well as their collaboration. In objectory, classes were divided into Entity classes ,interface classes and the control classes. The Computer Aided Software Engineering tools that are available commercially do not provide any assistance in this transition. CASE tools take advantage of meta modeling are helpful only after the construction of class diagram is completed.

In the Fusion method ,it used some object-oriented approaches like Object Modelling Technique(OMT),Class Responsibility Collaborator(CRC) and Objectory used the term Agents to represent some of the hardware and software systems .In Fusion method, there was no requirement phase ,where in a user will supply the initial requirement document. Any software project is worked out by both analyst and designer. The analyst creates the Use case diagram. The designer creates the Class diagram. But the designer can do this only after the analyst has created the Use case diagram. Once the design is over it is need to decide which software is suitable for the application.

The proposed system using content based image retrieval consists of 6 phases:

1. Getting input from user for indexing database images.
2. Calculating wavelet.
3. Clustering similar images.
4. Getting input from the user to search the image
5. Comparing the wavelets of input and database images
6. Displaying closest similar image

**Advantage:**

1. Once database indexed there is no need to index the database again

2. The user can search images without indexing the database again and again

3. Due to this there is least time required to search a give image

**The flow chart of the project is as follows**

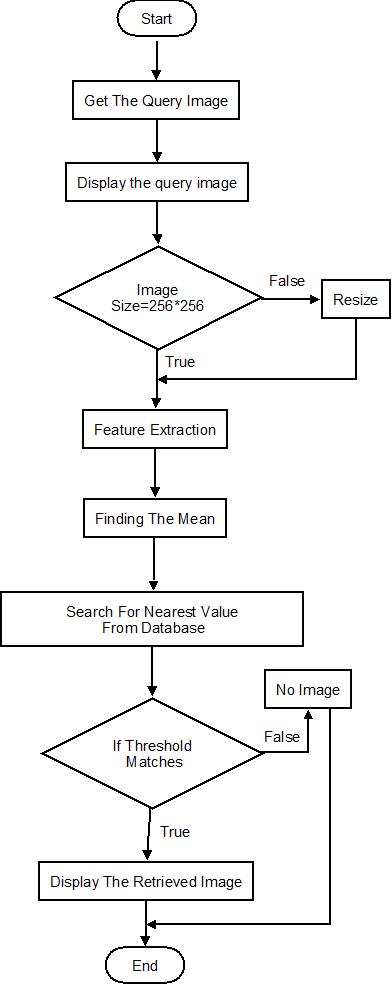
****

Fig 10:Flow Chart

**4.2 Module Description**

**MODULES**

* Getting Input from user to index the database images
* Calculating wavelet
* Clustering images
* Getting input from user to search the image
* Comparing the wavelets
* Displaying closest similar images

**Step1: Getting Input from user to index the database images**

User selects the folder for indexing images of database first.

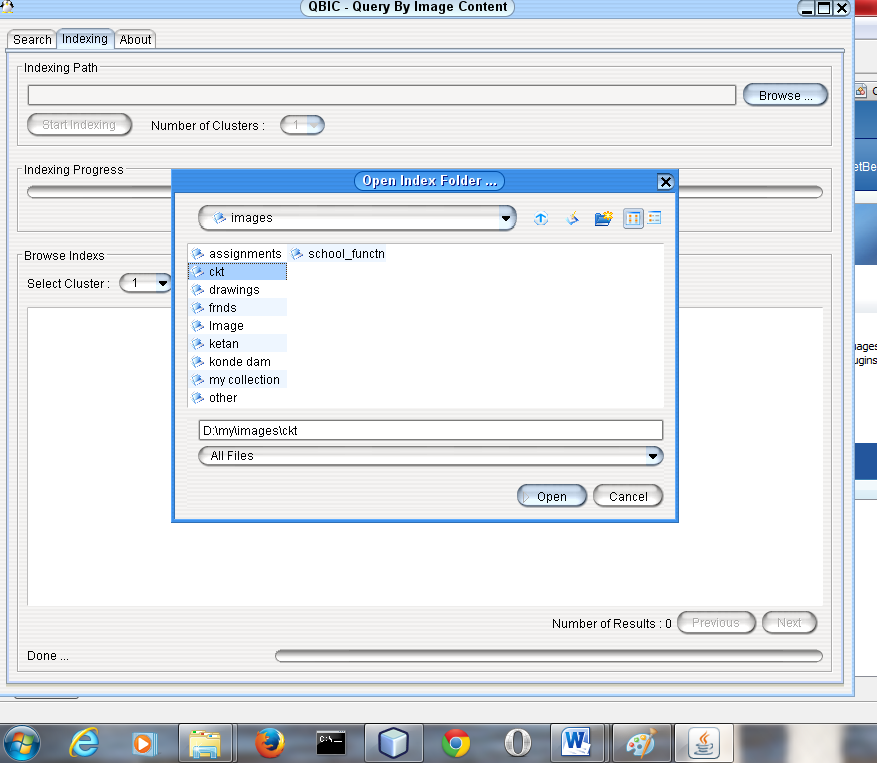


Fig11:Indexing Input

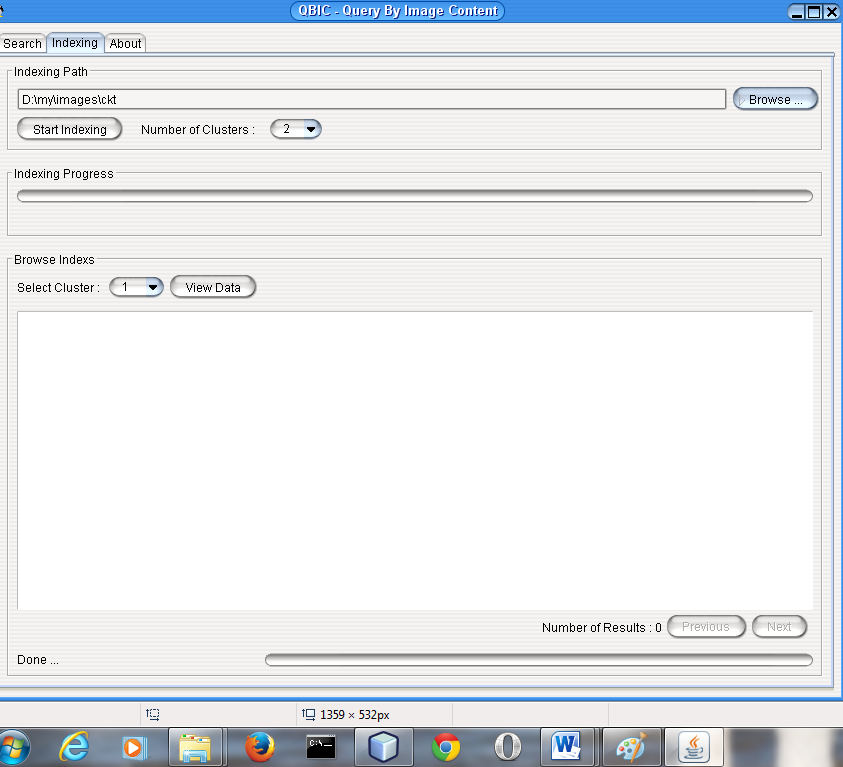


Fig12: Start Indexing

**Step2: Calculating wavelet**

Here we are going to calculate the wavelets of the images stored in the database i.e. we are going to extract the features from the database using haar wavelet transform. In haar wavelet transform we are going to select only those points which are enough to define the image rest of the points have been neglected.

The example is given below:

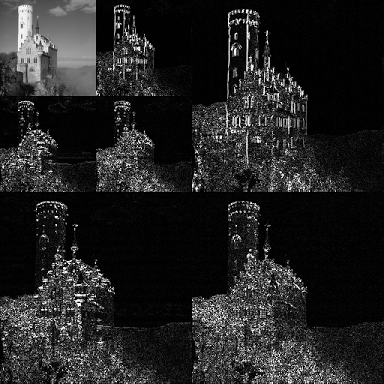


Fig 13: Calculated Wavelet

**Step3: Clustering images**

The closest and similar wavelet images are grouped i.e. put into similar clusters. Here we are going to apply K-means clustering algorithm. This algorithm helps in grouping similar images in the same cluster which may help in finding the related image in step 5. The one value represents the cluster which is the mean of all the stored values. This will help in reaching at the desired image while searching for the image which is stored in respective cluster. The clustered images are shown below:

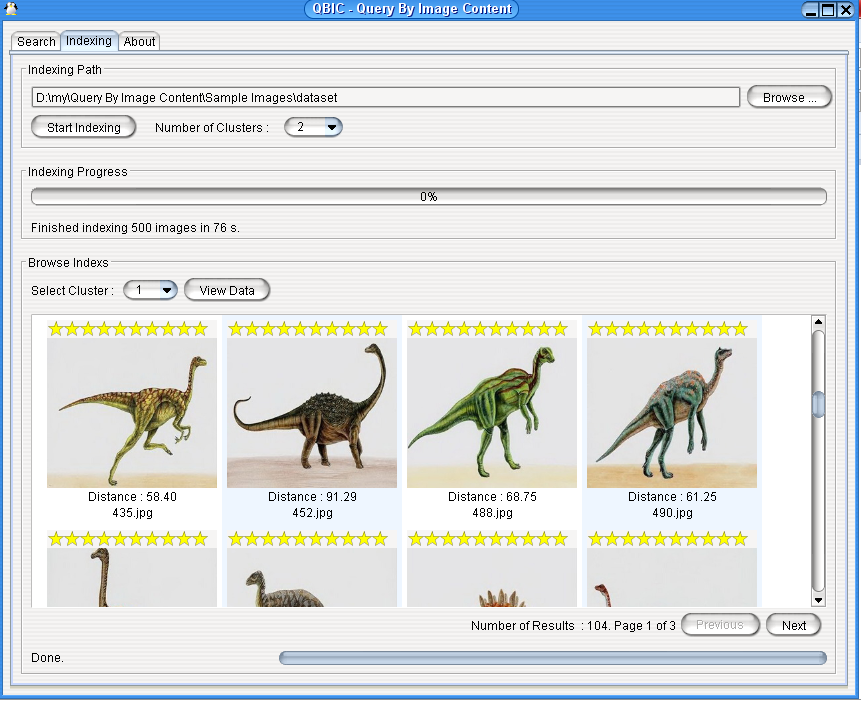


Fig 14: Clustered Images

**Step 4: Getting input from user to search the image**

Here we are going to take image as input which is we are going to search in our database.

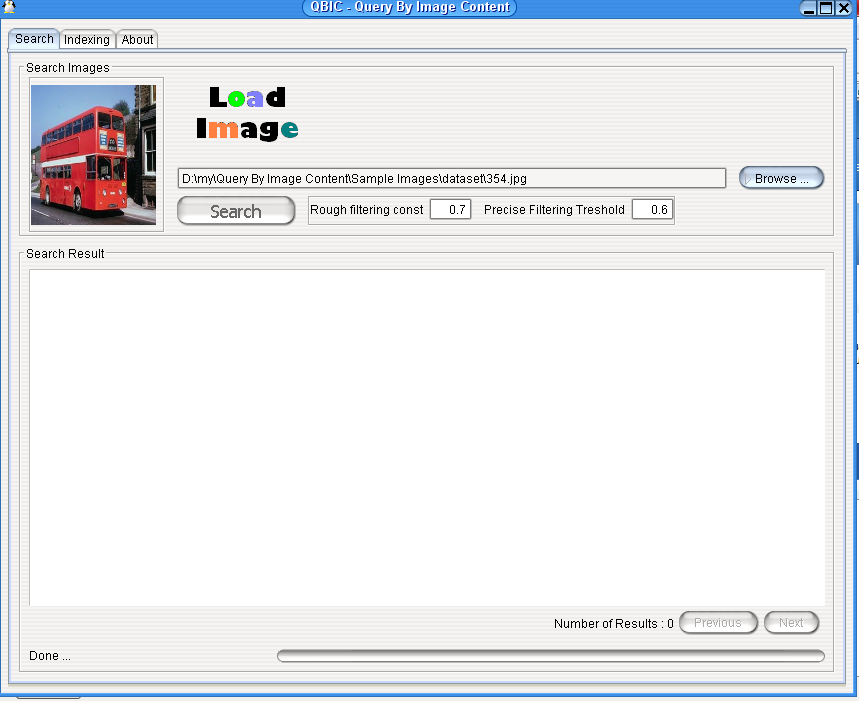


Fig 15: User input to search image

**Step5: Comparing the wavelets**

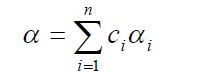
We repeat the step 2 to extract the features from the image. Then we compare the calculated value in step2 with all the means of the clusters with the help of Euclidian distance formula. Then we select the cluster which is having least distance with the input image. Then again with the help of Euclidian formula we are going to compare the distance between the input image and the images in the respective cluster.

* 1. **Similarity criteria**

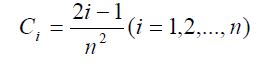
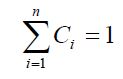
Define the Similarity αi of ΔAi and Δbi as:

C:\Users\Royal\Desktop\snip4.JPG

and we can thus give the similarity α of the matrices A and B as:



Where,

 and 

**Step6: Displaying closest similar images**

As per users input similar images are displayed.

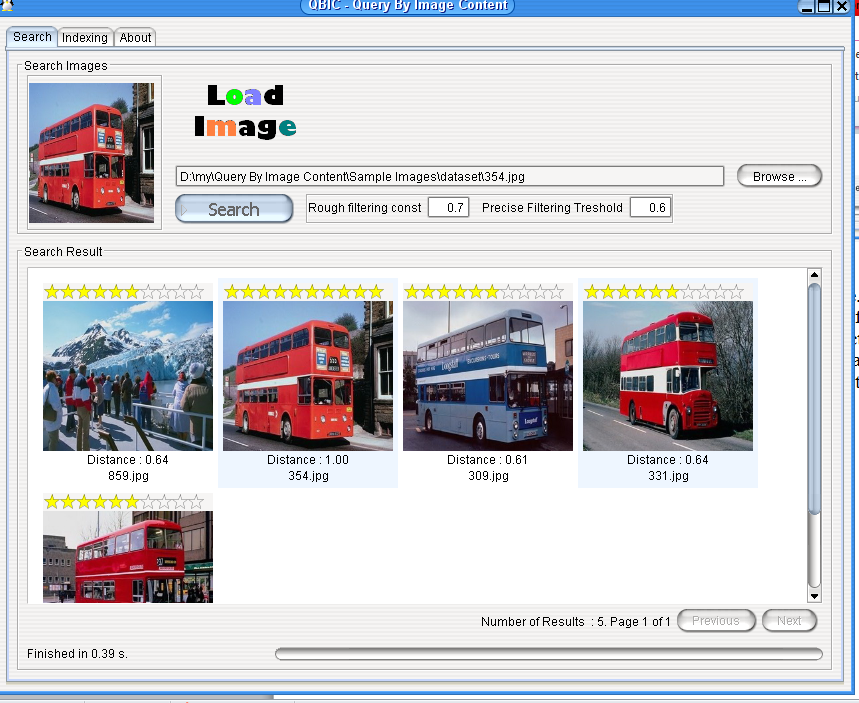
****

Fig 16: Output

**4.3 Design Details**

**Use Case diagram:**

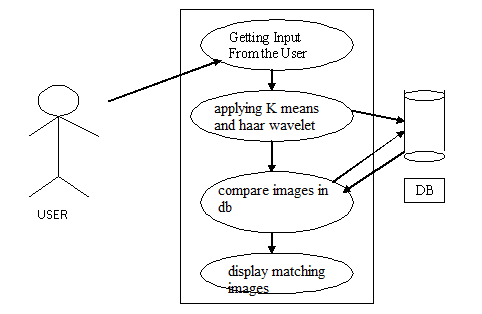


Fig 17: USE CASE DIAGRAM

**Data flow diagram:**

The DFD is also called as bubble chart.It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system,various processing carried out on these data,and the output data is generated by the system.

**DFD 0**

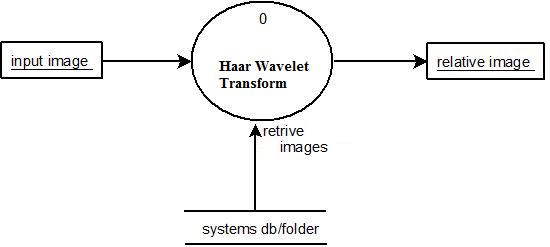
****

Fig 18: dFD 0

**DFD1**

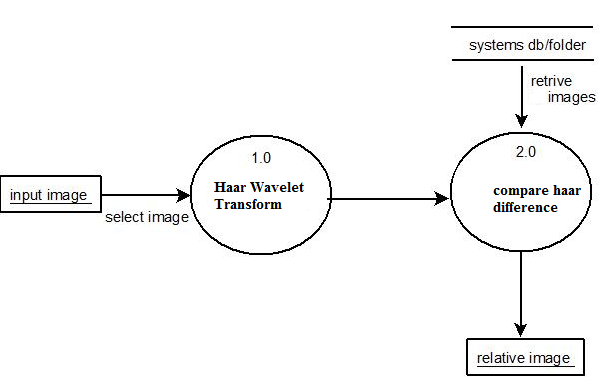
****

Fig 19: DFD 1

**DFD 1.1**

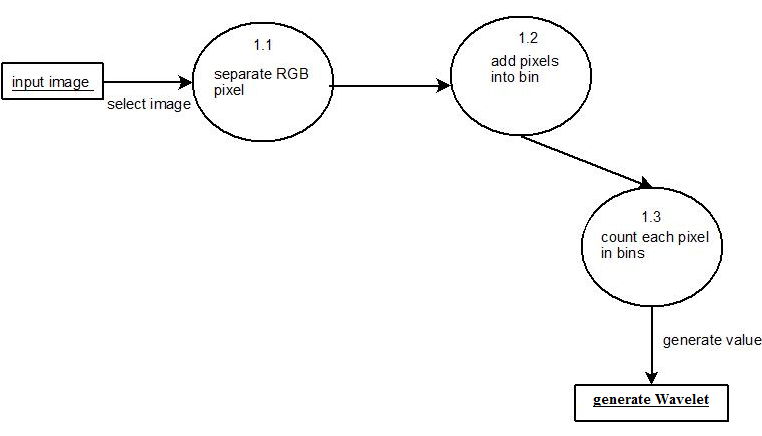
****

Fig20: DFD 1.1

**Activity Diagram**

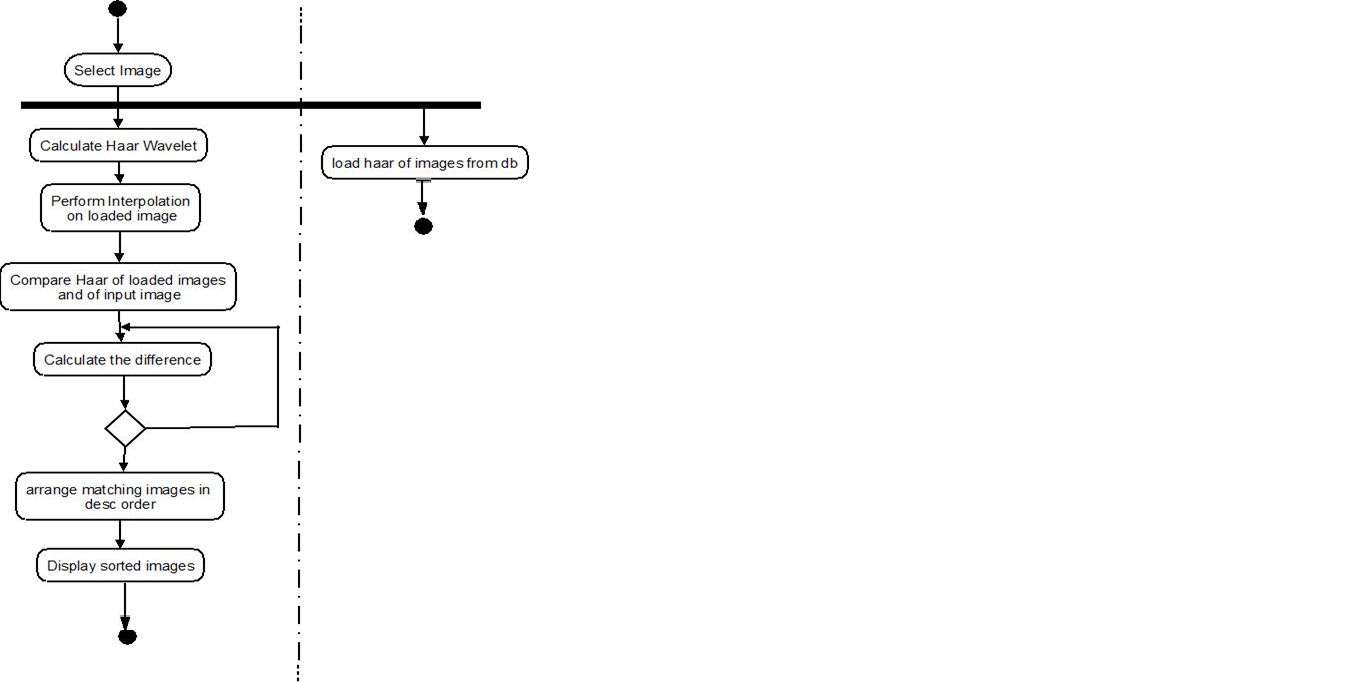
****

Fig 21Activity Diagram

**Chapter 4: Testing Technology**

**Testing Technology**

System testing is a critical phase implementation. Testing of the system involves hardware devise and debugging of the computer programs and testing information processing procedures. Testing can be done with text data, which attempts to stimulate all possible conditions that may arise during processing.

If structured programming methodologies have been adopted during coding the testing proceeds from higher level to lower level of program module until the entire program is tested as unit. The testing methods adopted during the testing of the system were unit testing and integrated testing.

**5.1 Unit Testing**

**Unit testing** is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. **Unit testing** is often automated but it can also be done manually.

Unit testing focuses first on the modules, independently of one another, to locate errors. This enables the tester to detect errors in coding and logical errors that is contained within that module alone. Those resulting from the interaction between modules are initially avoided.

**INTEGRATION TESTING:**

Integration testing is a systematic technique for constructing the program structure while at the same time to uncover the errors associated with interfacing. The objective is to take unit-tested module and build a program structure that has been detected by designing. It also tests to find the discrepancies between the system and its original objectives. Subordinate stubs are replaced one at time actual module. Tests were conducted at each module was integrated. On completion of each set another stub was replaced with the real module.

**5.2 SYSTEM TESTING**

**System testing** of software or hardware is **testing** conducted on a complete, integrated **system** to evaluate the **system's** compliance with its specified requirements. **System testing** falls within the scope of black box **testing**, and as such, should require no knowledge of the inner design of the code or logic.

**5.3 FUNCTIONAL TESTING:**

Functional testing is a technique in which all the functionalities of the program are tested to check whether all the functions that where proposed during the planning phases are full filled. This is also to check that if all the functions proposed are working properly.

This is further done in two phases:

* One before the integration to see if all the unit components work properly.
* Second to see if they still work properly after they have been integrated to check if some functional compatibility issues arise.

**5.4 PERFORMANCE TESTING:**

**Expected Result**

* The images stored in clusters should be according to k-means technique
* Related images should be properly displayed according to matching criteria.
* Images provided by the application should be correct and as per the user’s need.

**Observation**

* Clusters follow proper k-means algorithm.
* Related images are properly displayed according to matching criteria.
* Images coming from the database are correct.

**5.5 LOAD / STRESS TESTING :**

**Expected Result:**

* Response time should be unaffected irrespective of the no of images stored in database.
* The introduction of the newer images should not make the software to work hap hazardously.
* Continuous use of the software by user should not result into the software getting slowed down.

**Observation:**

* The speed of transmission was fine even when the newer images were getting added.
* The response of the software was satisfying even with the introduction of newer images in database.
  1. **Acceptance Testing:**

**Acceptance testing** is a test conducted to determine if the requirements of a [specification](http://en.wikipedia.org/wiki/Specification) or [contract](http://en.wikipedia.org/wiki/Contract) are met. It may involve [chemical tests](http://en.wikipedia.org/wiki/Chemical_test), [physical tests](http://en.wikipedia.org/wiki/Physical_test), or [performance tests](http://en.wikipedia.org/wiki/Performance_test_(assessment)). In [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering) it may involve [black-box testing](http://en.wikipedia.org/wiki/Black-box_testing) performed on a [system](http://en.wikipedia.org/wiki/System) (for example: a piece of [software](http://en.wikipedia.org/wiki/Software_system), lots of manufactured mechanical parts, or batches of chemical products) prior to its delivery.

**Test Cases:**

A test case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly. The process of developing test cases can also help find problems in the requirements or design of an application. A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

**Test Cases**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Item No** | | **Test Case name** | | **Step** | **Expected** | **Actual** | **Status** |
| 1 | | Index database images | | 1. select folder to index the images from that folder  2. Press index images button. | 1. System starts indexing the images | 1. System indexes the images from database using the haar transform and k-means algorithm | Pass |
| 2 | | Indexing unsuccessful due to absence of images in selected folder | | 1 select folder  2. Press start indexing button | 1. Software checks whether images exist in selected folder or not if not then select another folder”. | 1. software checks whether images present in database or not  2. the path inserted in software is kept blank | Pass |
| 3 | | Select the image to search related matching images | | 1. select the location of the image or insert the path of the image  2. Press the search button. | 1. software checks indexing is done or not 2. if not then deactivate search button | Software first checks indexing of database images is done or not if not then search image button is not activated | Pass |
| 4 | Unsuccessful selection of image due to wrong selection path | | Press the search button. | | 1. software checks at given location image exists or not if not then thumbnail of the image will not displayed and search button will not be activated | 1. software checks if image is exist of not if not then deactivates the search button | Pass |
| 5 | Successful selection of path of image which has to be searhed | | 1. select the proper path  2. Press the search image button. | | System gives proper images from indexed database which are closely matching to the image | System matches the images wiit related cluster and returns nearly matching images as an output | Pass |

Table 6.1: selecting and matching the image with indexed database

**Chapter 6 : Implementation Details**

**CHAPTER NO 6**

**IMPLEMENTATION DETAILS**

**6.1 Methodology:**

Block diagram of content based image retrieval using CBIR

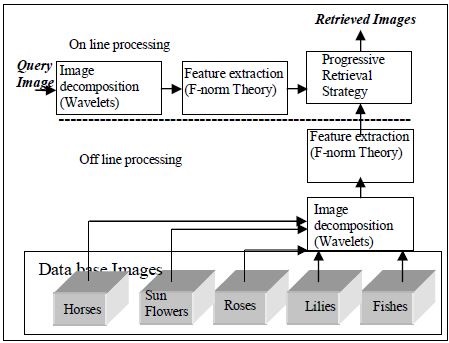


Fig 22: Block diagram for Content Based Image retrieval

At the first image is decomposed into different wavelets as we know image is the composition of wavelets. For the purpose of extracting the features from the image we are going to use haar wavelet transform and then these extracted features are stored in f normalised form. From all the extracted image features we are going to cluster the images based on the extracted coefficients with the help of k-means algorithm

* + 1. **Haar Wavelet Transform**

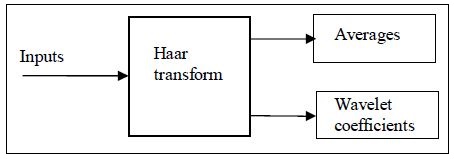
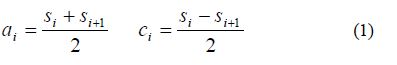
****

Fig.23: Haar Wavelet Transform

If a data set S0, S1 ….SN-1 contains N elements, there will be N/2 averages and N/2 wavelet coefficient values. The averages are stored in the upper half of the N element array and the coefficients are stored in the lower half as shown in the Fig.2. The averages become the input for the next step in the wavelet calculation, where for iteration i+1, Ni+1 = Ni/2. The recursive iterations continue until a single average and a single coefficient are calculated. This replaces the original data set of N elements with an average, followed by a set of coefficients whose size is an increasing power of two (Ex: 20, 21, 22 ... N/2).

The Haar equations to calculate an average *i a* and a wavelet coefficient *i c* from an odd and even element in the data set are:



Forward Haar transform for an eight element signal is shown in the Fig 3. Here signal is multiplied by the forward transform matrix.

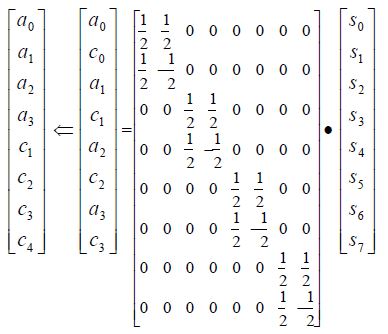


Fig 24: Haar Forward transform for 8 element signal

The arrow represents a split operation that reorders the result so that the average values are in the first half of the vector and the coefficients are in the second half. To complete the forward Haar transform there are two more steps. The next step would multiple the average values *i a* by a 4x4 transform matrix, generating two new averages and two new coefficients which would replace the averages in the first step. The last step would multiply these new averages by a 2x2 matrix generating the final average and the final coefficient.

Haar Wavelet transform is shown below:

**Wavelet.java**

import org.ejml.data.DenseMatrix64F;

public abstract class Wavelet {

public DenseMatrix64F [] transform(DenseMatrix64F m, int n) {

DenseMatrix64F matrix = m.copy();

DenseMatrix64F [] matrixs = new DenseMatrix64F[n];

if (matrix.getNumCols() != matrix.getNumRows()) {

throw new RuntimeException("Matrix must be square!");

}

for (int j = 0; j < n; j++) {

for (int c = 0; c < matrix.getNumCols(); c++) {

double[] col = new double[matrix.getNumRows()];

for (int rows = 0; rows < matrix.getNumRows(); rows++) { for (int cols = c; cols < c + 1; cols++) {

col[rows] = matrix.get(rows, cols);

}

}

col = forwardStep(col);

for (int r = 0; r < matrix.getNumRows(); r++) {

matrix.set(r, c, col[r]);

}

}

for (int r = 0; r < matrix.getNumRows(); r++) {

double[] row = new double[matrix.getNumCols()];

for (int rows = r; rows < r + 1; rows++) {

for (int cols = 0; cols < matrix.getNumCols(); cols++) {

row[cols] = matrix.get(rows, cols);

}

}

row = forwardStep(row);

for (int c = 0; c < matrix.getNumCols(); c++) {

matrix.set(r, c, row[c]);

}

}

matrixs[j] = matrix;

}

return matrixs;

}

public DenseMatrix64F [] transform(DenseMatrix64F matrix) {

return transform(matrix, BinaryOps.log2(matrix.getNumRows()));

}

protected abstract double[] forwardStep(double[] signal);

}

**Haar.java**

public class HaarWavelet extends Wavelet {

private static double coeff = 2;

@Override

protected double[] forwardStep(double[] values) {

int num\_c = values.length / 2;

double[] result = new double[num\_c \* 2];

for (int i = 0, j = 0; i < values.length; i += 2, j++) {

result[j] = (values[i] + values[i + 1]) / coeff;

result[num\_c + j] = (values[i] - values[i + 1]) / coeff;

}

return result;

}

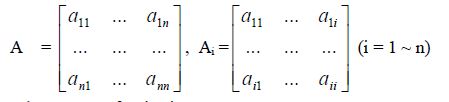
}

* + 1. **Feature Extraction and Similarity criteria**

Our QBICalgorithm is based on direct wavelet decomposition of image in *RGB* color space and utilizes the “query by example” method. With approaches mentioned above, database images are decomposed offline into multi-level coefficients from -1 to -J levels, with which, we can generate colour feature database and perform similarity match between images. After decomposition, each resulting sub image is in fact a coefficient matrix, where, by special processing, large coefficients with more energy can be distributed in the upleft area, therefore, with F-norm theory, we can well decrease the dimension of image feature and perform highly efficient image matching.

* + 1. **Feature Vector**

Suppose A is a square matrix and Ai is its ith order sub matrix where

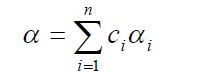


* + 1. **Similarity criteria**

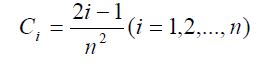
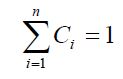
Define the Similarity αi of ΔAi and ΔBi as:

snip4

and we can thus give the similarity α of the matrices A and B as:



Where,

 and 

While searching for the image we are going to search image with help of following program:

import sp.code.qbic.Constants;

import sp.code.qbic.clustering.Cluster;

import sp.code.qbic.indexing.Index;

import sp.code.qbic.featureextraction.FNormFeature;

import sp.code.qbic.indexing.ClusterReader;

import sp.code.qbic.indexing.IndexReader;

import sp.code.qbic.util.GraphicsUtilities;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import javax.imageio.ImageIO;

import org.ejml.data.DenseMatrix64F;

public class ImageSearcher {

private FNormFeature feature = new FNormFeature();

public List<ImageResult> search(BufferedImage queryImage, double alpha, double beta, boolean useCluster) throws IOException {

IndexReader indexReader = new IndexReader();

indexReader.open();

List<Cluster> clusters = indexReader.getClusters();

indexReader.close();

List<Index> indexes = new ArrayList<Index>();

ClusterReader clusterReader = new ClusterReader();

Index queryIndex = clusterReader.getIndex(queryImage);

if (useCluster) {

int cid = -1;

double dist = 0;

double prevDist = 65280;

for (int c = 0; c < clusters.size(); c++) {

dist = clusters.get(c).getDistance(queryIndex);

if (dist < prevDist) {

cid = c;

prevDist = dist;

}

}

if (cid > -1) {

clusterReader.open(cid);

indexes.addAll(clusterReader.getIndexes());

clusterReader.close();

System.out.println(indexes.size());

}

} else {

for (int c = 0; c < clusters.size(); c++) {

clusterReader.open(c);

indexes.addAll(clusterReader.getIndexes());

clusterReader.close();

}

}

if (indexes.isEmpty()) {

return Collections.EMPTY\_LIST;

}

List<Index> filteredIndexes = progressiveRetrievalStrategy(indexes, queryIndex, clusterReader,

alpha, beta);

return rankedSimilarity(queryIndex, clusterReader, filteredIndexes);

}

private List<Index> progressiveRetrievalStrategy(List<Index> indexes, Index queryIndex, ClusterReader reader,

double alpha, double beta) {

List<Index> filteredIndexs = new ArrayList<Index>();

double[] RLLVarianceQ = null;

double[] GLLVarianceQ = null;

double[] BLLVarianceQ = null;

double[] RLHVarianceQ = null;

double[] GLHVarianceQ = null;

double[] BLHVarianceQ = null;

double[] RHLVarianceQ = null;

double[] GHLVarianceQ = null;

double[] BHLVarianceQ = null;

DenseMatrix64F[] RLLFeatureQ = null;

DenseMatrix64F[] GLLFeatureQ = null;

DenseMatrix64F[] BLLFeatureQ = null;

RLLVarianceQ = reader.toDoubles(queryIndex.getRLLhaarvariance());

GLLVarianceQ = reader.toDoubles(queryIndex.getGLLhaarvariance());

BLLVarianceQ = reader.toDoubles(queryIndex.getBLLhaarvariance());

RLHVarianceQ = reader.toDoubles(queryIndex.getRLHhaarvariance());

GLHVarianceQ = reader.toDoubles(queryIndex.getGLHhaarvariance());

BLHVarianceQ = reader.toDoubles(queryIndex.getBLHhaarvariance());

RHLVarianceQ = reader.toDoubles(queryIndex.getRHLhaarvariance());

GHLVarianceQ = reader.toDoubles(queryIndex.getGHLhaarvariance());

BHLVarianceQ = reader.toDoubles(queryIndex.getBHLhaarvariance());

RLLFeatureQ = reader.toMatrixs(queryIndex.getRLLhaarfeature());

GLLFeatureQ = reader.toMatrixs(queryIndex.getGLLhaarfeature());

BLLFeatureQ = reader.toMatrixs(queryIndex.getBLLhaarfeature());

for (Index index : indexes) {

if (progressive(reader, alpha, beta, index,

RLLVarianceQ, GLLVarianceQ, BLLVarianceQ,

RLHVarianceQ, GLHVarianceQ, BLHVarianceQ,

RHLVarianceQ, GHLVarianceQ, BHLVarianceQ,

RLLFeatureQ, GLLFeatureQ, BLLFeatureQ)) {

filteredIndexs.add(index);

}

}

return filteredIndexs;

}

private boolean progressive(ClusterReader reader, double alpha, double beta, Index index, double[] rllvarQ,

double[] gllvarQ, double[] bllvarQ, double[] rlhvarQ,

double[] glhvarQ, double[] blhvarQ, double[] rhlvarQ,

double[] ghlvarQ, double[] bhlvarQ,

DenseMatrix64F[] rllfeatureQ, DenseMatrix64F[] gllfeatureQ,

DenseMatrix64F[] bllfeatureQ) {

double[] RLLVarianceD = reader.toDoubles(index.getRLLhaarvariance());

double[] GLLVarianceD = reader.toDoubles(index.getGLLhaarvariance());

double[] BLLVarianceD = reader.toDoubles(index.getBLLhaarvariance());

double[] RLHVarianceD = reader.toDoubles(index.getRLHhaarvariance());

double[] GLHVarianceD = reader.toDoubles(index.getGLHhaarvariance());

double[] BLHVarianceD = reader.toDoubles(index.getBLHhaarvariance());

double[] RHLVarianceD = reader.toDoubles(index.getRHLhaarvariance());

double[] GHLVarianceD = reader.toDoubles(index.getGHLhaarvariance());

double[] BHLVarianceD = reader.toDoubles(index.getBHLhaarvariance());

DenseMatrix64F[] RLLFeatureD = reader.toMatrixs(index.getRLLhaarfeature());

DenseMatrix64F[] GLLFeatureD = reader.toMatrixs(index.getGLLhaarfeature());

DenseMatrix64F[] BLLFeatureD = reader.toMatrixs(index.getBLLhaarfeature());

for (int j = Constants.DECOMPOSITION\_LEVEL - 1; j >= 0; j--) {

if (!roughFilter(beta, rllvarQ[j], gllvarQ[j], bllvarQ[j], RLLVarianceD[j],

GLLVarianceD[j], BLLVarianceD[j])) {

return false;

}

if (!roughFilter(beta, rlhvarQ[j], glhvarQ[j], blhvarQ[j], RLHVarianceD[j],

GLHVarianceD[j], BLHVarianceD[j])) {

return false;

}

if (!roughFilter(beta, rhlvarQ[j], ghlvarQ[j], bhlvarQ[j], RHLVarianceD[j],

GHLVarianceD[j], BHLVarianceD[j])) {

return false;

}

if (preciseFilter(alpha, rllfeatureQ[j], gllfeatureQ[j], bllfeatureQ[j],

RLLFeatureD[j], GLLFeatureD[j], BLLFeatureD[j])) {

return false;

}

}

return true;

}

private boolean roughFilter(double beta, double sigmaRQ, double sigmaGQ, double sigmaBQ,

double sigmaRD, double sigmaGD, double sigmaBD) {

return ((beta \* sigmaRQ) < sigmaRD)

&& (sigmaRD < (sigmaRQ / beta))

&& ((beta \* sigmaGQ) < sigmaGD)

&& (sigmaGD < (sigmaGQ / beta))

&& ((beta \* sigmaBQ) < sigmaBD)

&& (sigmaBD < (sigmaBQ / beta));

}

private boolean preciseFilter(double alpha, DenseMatrix64F rq, DenseMatrix64F gq, DenseMatrix64F bq,

DenseMatrix64F rd, DenseMatrix64F gd, DenseMatrix64F bd) {

double redSimilarity = feature.calculateSimilarity(rq, rd);

double greenSimilarity = feature.calculateSimilarity(gq, gd);

double blueSimilarity = feature.calculateSimilarity(bq, bd);

double similarity = (redSimilarity + greenSimilarity + blueSimilarity) / 3;

return similarity < alpha;

}

public List<ImageResult> rankedSimilarity(Index queryIndex, ClusterReader reader,

List<Index> filteredIndexs) {

List<ImageResult> imageResults = new ArrayList<ImageResult>();

DenseMatrix64F redFeatureA = null;

DenseMatrix64F greenFeatureA = null;

DenseMatrix64F blueFeatureA = null;

redFeatureA = reader.toMatrix(queryIndex.getRHaarFeature());

greenFeatureA = reader.toMatrix(queryIndex.getGHaarFeature());

blueFeatureA = reader.toMatrix(queryIndex.getBHaarFeature());

for (Index index : filteredIndexs) {

DenseMatrix64F redFeatureB = null;

DenseMatrix64F greenFeatureB = null;

DenseMatrix64F blueFeatureB = null;

redFeatureB = reader.toMatrix(index.getRHaarFeature());

greenFeatureB = reader.toMatrix(index.getGHaarFeature());

blueFeatureB = reader.toMatrix(index.getBHaarFeature());

double redSimilarity = feature.calculateSimilarity(redFeatureA, redFeatureB);

double greenSimilarity = feature.calculateSimilarity(greenFeatureA, greenFeatureB);

double blueSimilarity = feature.calculateSimilarity(blueFeatureA, blueFeatureB);

double similarity = (redSimilarity + greenSimilarity + blueSimilarity) / 3;

ImageResult imageResult = new ImageResult();

imageResult.setPath(index.getFilePath());

imageResult.setSimilarity(similarity);

try {

BufferedImage bufferedImage = ImageIO.read(new File(index.getFilePath()));

imageResult.setBufferedImage(GraphicsUtilities.resizeImage(bufferedImage, Constants.THUMB\_WIDTH,

Constants.THUMB\_HEIGHT));

imageResults.add(imageResult);

} catch (IOException ex) {

}

}

Collections.sort(imageResults);

return imageResults;

}

}

* 1. **Progressive Retrieval Strategy**

We use progressive retrieval strategy in order to balance between computational complexity and retrieval accuracy.

* + 1. **Rough filtering:**

Starting from the maximal decomposition level –J, With the resulting LL coefficients, we calculate standard variances vectors of the query image and the database image *as (σrq, σgq,, σbq*) and (*σrd, σgd, σbd*) respectively, after which,we can roughly filter database images as follows:

F= (βσrq < σrd< σrq /β) && (βσgq < σgd < σgq /β)&& (βσbq < σbd < σbq/β)

Where, the filtering constant β Є (0, 1) is used to adjust the sifted database images. If F is false, then database image can be identified as far apart from Query image and therefore is discarded; else, database image be kept

for further match.

* + 1. **Progressive rough filtering:**

Considering the effect of high frequency components with LH and HL wavelet coefficients in previous step.

* + 1. **More precise filtering:**

With the obtained LL coefficients, which best reflect the general feature of image; we use the similarity criteria to determine more precise images. If α exceeds a given threshold, it means that mismatch occurs and it should be discarded; else, it will be kept for further match.

* + 1. **Iteration:**

J=J-1, and iterate step 1~ 4 till J = 0. Finally, retrieval results are fed back to users in the order of their similarity values.

**6.3 Clustering Images:**

Clustering is an unsupervised task that can divide the given set of data into several non-overlapping homogenous groups. Each group, called cluster, consists of objects that are similar between themselves and dissimilar to objects of other groups. Clustering technique is extensively used in various applications in engineering, statistics and [**numerical analysis**](http://www.scialert.net/asci/result.php?searchin=Keywords&cat=&ascicat=ALL&Submit=Search&keyword=numerical+analysis). The k-means algorithm is by far the most popular clustering tool used in scientific and industrial applications. The name comes from representing each of K clusters Sj by the mean (or weighted average) siof its points, called as the center of the cluster or the centroid. The sum of the Euclidian distance d (x,y) between a point (xi) and its centroid (sj) is used as an objective function. The k-means algorithm partitions the data to minimize the criterion:

image1-2k7-148-153

**Basic K-means algorithm:** The basic steps involved in K-means clustering is outlined as:

|  |  |
| --- | --- |
| • | Select K-data points as the initial centroids. |

|  |  |
| --- | --- |
| • | (Re) assign all points to their closest centroids. |

|  |  |
| --- | --- |
| • | Recompute the centroid of each newly assembled cluster. |

|  |  |
| --- | --- |
| • | Repeat steps 2 and 3 until the centroids do not change. |

**Algorithm can be implemented in code as given below:**

**ClusterWriter.java**

public class ClusterWriter extends Indexer {

private org.apache.lucene.index.IndexWriter writer;

public void open(int id) throws CorruptIndexException, LockObtainFailedException, IOException {

writer=new org.apache.lucene.index.IndexWriter(FSDirectory.open(new File(Constants.CLUSTER\_IDX+"-"+id)),

new SimpleAnalyzer(),

true, org.apache.lucene.index.IndexWriter.MaxFieldLength.UNLIMITED);

}

public void addIndex(Index index) throws Exception{

Document document = new Document();

Method[] methods = Index.class.getDeclaredMethods();

for (Method m : methods) {

String methodName = m.getName();

if (methodName.startsWith("get")) {

String value = (String) m.invoke(index, new Object[]{});

Field f = new Field(methodName.replaceFirst("get", ""), value, Field.Store.YES, Field.Index.NO);

document.add(f);

}

}

writer.addDocument(document);

}

public void close() throws CorruptIndexException, IOException {

writer.optimize();

writer.close();

}

}

**Chapter 7: System Requirements**

**Chapter 7**

**System Requirements**

**Hardware Specification**

* Hard disk : 40 GB
* RAM : 512 MB
* Processor Speed : 3.00GHz
* Processor : Pentium IV Processor

**Software Specification**

* JAVA (JDK)
* Netbeans
* Swing Builder
* MS-Access

**Chapter 8:**

**Conclusion and Future Scope**

**Chapter 8**

**CONCLUSION AND FUTURE SCOPE**

**7.1 Conclusion**

We have presented a brief survey on work related to the young and exciting ﬁelds of content-based image retrieval and automated image annotation, spanning 120 publications in the current decade. We believe that the ﬁeld will experience a paradigm shift in the foreseeable future, with the focus being more on application-oriented, domain-speciﬁc work, generating considerable impact in day-to-day life. We have laid out some guidelines for building practical, real-world systems that we perceived during our own implementation experiences. Finally, we have compiled research trends in CBIR and automated annotation using Google Scholar’s search tool and citation scores. The trends indicate that while systems, feature

extraction, and relevance feedback have received a lot of attention, application-oriented aspects such as interface, visualization, scalability, and evaluation have traditionally

received lesser consideration. We feel that for all practical purposes, these aspects should also be considered equally important. Meanwhile, the quest for robust and reliable image understanding technology needs to continue as well. The future of this ﬁeld depends on the collective focus and overall progress in each aspect of image retrieval, and how

much the ordinary individual stands to beneﬁt from it.

.**7.2 Future Scope**

* In future we would like add more features and make more user- friendly
* We would like to make more accurate and precise and have a high end input system.
* It should be merged with more complex systems.
* It should also be more enhanced regarding the security issues.

**9.Bibliography**

**Chapter 9**

**Bibliography**

1. [http://en.wikipedia.org/wiki/Content\_based\_image\_retrieval]. Introduction and concepts
2. [http http://www.eecs.berkeley.edu/~cchang/docs/Spie01.pdf] content based image retrieval by Bhakthvatsalam (ppt) – technique used, reference feedback.
3. content-based Image RetrievalJohn Eakins Margaret Graham University of Northumbria at Newcastle [Technique used]
4. Image retrieval based on region shape similarity by Cheng Chang , Liu Wenyin and Hongjiang Zhang.[22-25]
5. Feature extraction and image retrieval by Mark s Nixon and alberto 1st edition[99-157].
6. . http://homepages.inf.ed.ac.uk/rbf/HIPR2/gryimage.html [gray scale images].
7. Issues [http://imageprocessing.wordpress.com/2007/12/09/content-based-image-retrieval-cbir/]
8. Color & Texture Feature Extraction for Content Based Image Retrieval Nitin Jain & Dr. S. S. Salankar\* Assistant Professor, Lokmanya Tilak College of Engineering, Navi Mumbai, India. IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)e-ISSN: 2278-1676, p-ISSN: 2320-3331
9. Content Based Image Retrieval – A Literature ReviewGaurav Jaswal(student), Amit KaulNational Institute of Technology, Hamirpur-177001, Himachal Pradesh(India)National Conference on Computing, Communication and Control (CCC‐09)
10. Image Retrieval: Ideas, Influences, and Trends of the New Age RITENDRA DATTA, DHIRAJ JOSHI, JIA LI, and JAMES Z. WANG The Pennsylvania State University
11. A Survey: Content Based Image Retrieval Based On Color, Texture, Shape & Neuro Fuzzy Mohd. Danish\*, Ritika Rawat \*\*, Ratika Sharma \*\*\* \*(Department of CSE A.I.M.T Greater Noida, India)\*\*(Department of CSE A.I.M.T Greater Noida, India) \*\*\*(Department of CSE A.I.M.T Greater Noida, India) Mohd. Danish et al. Int. Journal Of Engineering Research And Applications ISSN : 2248-9622, Vol. 3, Issue 5, Sep-Oct 2013, Pp.839-84
12. Image Retrieval: Current techniques, promising Directions and Open issues Young Rui and Thomas S.Huang Dept of ECE & Beckman Institute University Illinois at Urbaba-Compaign, Urbana and Shih-Fu Chang Dept of EE & New Media technology Center Columbia University New York
13. CSE 494 CSE/CBS 598 (Fall 2007): Numerical Linear Algebra for Data Exploration| Clustering Instructor: Jieping Ye
14. Color Image Segmentation Using K-Means Clustering and Otsu’s Adaptive Thresholding Vijay Jumb, Mandar Sohani, Avinash Shrivas
15. Content based Image Retrieval Review on its Methods and Transforms Avanish Tiwari and Anurag Jain Department of Computer Science and Engineering, Radharaman Institute of Technology and Science, Bhopal, Madhya Pradesh, India
16. Enhancing Content-based Image Retrieval using Moving K-Means Clustering Algorithm Shefalli and Balkrishan Jindal Research Scholar Yadavindra College of Engineering,Guru Kashi Campus, Punjabi University, Talwandi Sabo, Punjab, India
17. Content Based Image Retrieval: Survey Mehwish Rehman, Muhammad Iqbal, Muhammad Sharif and Mudassar Raza COMSATS Institute of Information Technology, Wah Cantt, Pakistan