International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 10 Issue 6 ǁ 2022 ǁ PP. 1086-1105

Image Classification in Content-Based Image Retrieval Systems

Dr.S.ERANA VEERAPA DINESH M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR
DEPT OF COMPUTER SCIENCE & ENGINEERING
P.S.R.R COLLEGE OF ENGINEERING
SIVAKASI -626140

Mrs.K. R INDIRA M.E.,
SUPERVISOR
PROFESSOR
DEPT OF COMPUTER SCIENCE & ENGINEERING
P.S.R.R COLLEGE OF ENGINEERING
SIVAKASI-626140

ABSTRACT
Multimedia contents are growing explosively and the need for multimedia retrieval is occurring more and more frequently in our daily life. Due to the complexity of multimedia contents, image understanding is a difficult but interesting issue in this field. Extracting valuable knowledge from a large-scale multimedia repository, so-called multimedia mining, has been recently studied by some researchers. Relevance feedback is a feature of some information retrieval systems. The idea behind relevance feedback is to take the results that are initially returned from a given query and to use information about whether or not those results are relevant to perform a new query. Relevance Feedback treats the retrieval session as repetitive query reformulation operation. Through successive human-computer interaction, the query descriptive information (features, matching models, metrics and any meta-knowledge) is repeatedly modified as a response to the user's feedback on retrieved results. Although a number of Relevance Feedback have been made on interactive CBIR, they still incur some common problems, namely redundant browsing and exploration convergence. First, in terms of redundant browsing, most existing RF methods focus on how to earn the user's satisfaction in one query process.

KEYWORDS:- Multimedia Retrieval, Image Classification, Image Retrieval System Using Artificial Intelligence

---

Date of Submission: 02-06-2022 Date of acceptance: 16-06-2022

I. INTRODUCTION

INTRODUCTION

Due to recent development in technology, there is an increase in the usage of digital cameras, smartphone, and Internet. The shared and stored multimedia data are growing, and to search or to retrieve a relevant image from an archive is a challenging research problem. The fundamental need of any image retrieval model is to search and arrange the images that are in a visual semantic relationship with the query given by the user. Most of the search engines on the Internet retrieve the images on the basis of text-based approaches that require captions as input.

Content-based image retrieval **(CBIR)** is a framework that can overcome the problems as it is based on the visual analysis of contents that are part of the query image. To provide a query image as an input is the main requirement of CBIR and it matches the visual contents of query image with the images that are placed in the archive, and closeness in the visual similarity in terms of image feature vector provides a base to find images with similar contents. For example in a great database the images can be divided into such classes as follows: landscapes, buildings, animals, faces, artificial images, etc. Many color image classification methods use color histograms.The aim of this paper to develop such color histogram based classification approach, which is efficient, quick and enough robust. In the interest of this I used some features of color histograms, and classified the images using these features.The advantage of this approach is the comparison of histogram features is much faster and more efficient than of other commonly used methods. In content-based image retrieval systems (CBIR) the most efficient and simple searches are the color based searches. Although this methods can be improved if some preprocessing steps are used. In this paper one of the preprocessing algorithms, the image classification is analyzed. In CBIR image classification has to be computationally fast and efficient. In this paper a new approach is introduced, which based on low level image histogram features. The main advantage of this method is the very quick generation and comparison of the applied features vectors.

II. OVERVIEW OF PROJECT

Content-based image retrieval uses the visual contents of an image such as color, shape, texture, and spatial layout to represent and index the image. In typical content-based image retrieval systems, the visual contents of the images in the database are extracted and described by multi-dimensional feature vectors. The feature vectors of the images in the database form a feature database. To retrieve images, users provide the retrieval system with example images or sketched figures. The system then changes these examples into its internal representation of feature vectors. The similarities /distances between the feature vectors of the query example or sketch and those of the images in the database are then calculated and retrieval is performed with the aid of an indexing scheme.

The indexing scheme provides an efficient way to search for the image database. Recent retrieval systems have incorporated users' relevance feedback to modify the retrieval process in order to generate perceptually and semantically more meaningful retrieval results. In this chapter, we introduce these fundamental techniques for content-based image retrieval.

In early text-based image retrieval methods can be found in. Text-based image retrieval uses traditional database techniques to manage images. Through text descriptions, images can be organized by topical or semantic hierarchies to facilitate easy navigation and browsing based on standard Boolean queries. However, since automatically generating descriptive texts for a wide spectrum of images is not feasible, most text-based image retrieval systems require manual annotation of images. Obviously, annotating images manually is a cumbersome and expensive task for large image databases, and is often subjective, context-sensitive and incomplete. As a result, it is difficult for the traditional text-based methods to support a variety of task-dependent queries.

The project involves the following modules:

MODULE 1

➢ **Image as input:**

This module gets the input image from the user and sends the query image to the histogram calculation part. This is developed by MVC architecture, so the security, calculation and processing section is become very easier.

MODULE 2

➢ **Local Histogram:**

The histogram of an image is a plot of the gray level values
or the intensity values of a color channel versus the number of pixels at that value. The features based on the first order histogram probability are the mean, standard deviation, skew, energy, and entropy.

MODULE 3

➢ **Global Histogram:**

To perform the classification we need methods to compare two feature vectors. The primary methods are to either measure the difference between the two, or to measure the similarity. Two vectors that are closely related will have a small difference and a large similarity. Quadratic distance is used for measuring the distance between two vectors.

MODULE 4

➢ **Extracting the relevant images:**

The final system is a jsp, servlet based application which serves as an interface to feed in the contents and control instructions which is interpreted on the server and the appropriate action is taken. The output has the task of displaying image contents. The output creation task is done through a data entry interface which contains various sections to be filled.

DEVELOPMENT ENVIRONMENT

OVERVIEW OF JAVA

Java is an Object oriented application programming language developed by Sun Microsystems. Java is a very powerful general-purpose programming language. It is a stupendous programming language which is not confined to machine applications only. Java is more than just a tool for building transportable multimedia applications. Due to its versatility, it is a platform independent language, be it a hardware platform or any operating system.

Java programs run as quickly and efficiently as C++ programs due to the implementation of the JVM (Java Virtual Machine). It adds to C++ in the areas of automatic memory management. Moreover it also extends the language support for multithreaded applications. The Java applet API is a framework that allows Java-enabled Web browsers to manage and display embedded Java applications within.

Java applications can be executed by any software that implements the Java run-time system. Any kind of applications can be written in Java programming language such as any small or large applications, or any standalone application.

There are four main pillars of an Object Oriented Programming Language:

- **INHERITANCE**

Inheritance provides the facility to drive one class by another using simple syntax. You can say that it is a process of creating new class and use the behavior of the existing class by extending them for reuse the existing code and adding the additional features as you need. It also used to manage and make well structured software.

- **ENCAPSULATION**

Encapsulation is the ability to bundle the property and method of the object and also operate them. It is the mechanism of combining the information and providing the abstraction as well.

- **POLYMORPHISM**

As the name suggest one name multiple form, Polymorphism is the way that provide the different functionality by the functions having the same name based on the signatures of the methods. There are two type of polymorphism first is run-time polymorphism and second is compile-time polymorphism.

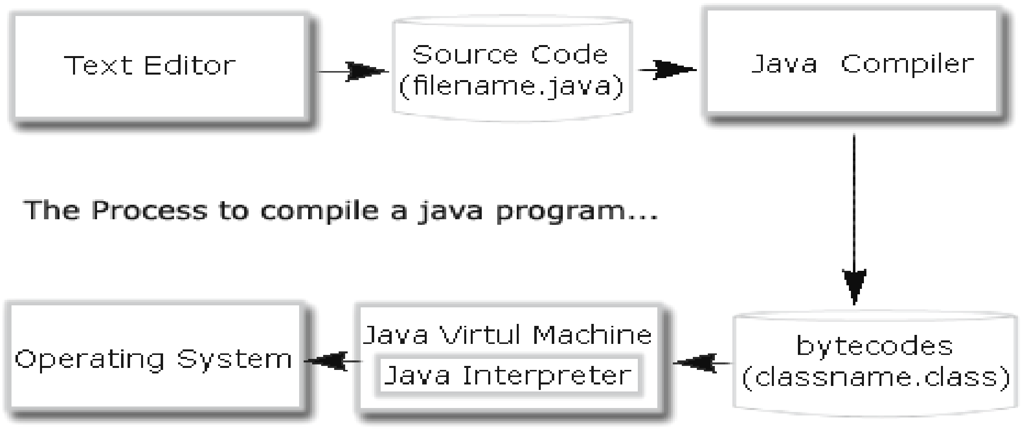
- **DYNAMIC BINDING**

It is the way that provides the maximum functionality to a program for a specific type at runtime. There are two type of binding first is dynamic binding and second is static binding.

JAVA TOOLS

➢ **COMPILER**

A Java Compiler java is a computer program or set of programs which translates java source code into java byte code. The output from a Java compiler comes in the form of Java class files (with .class extension). The java source code contained in files end with the .java extension. The file name must be the same as the class name, as classname.java. When the javac compiles the source file defined in a .java files, it generates byte code for the java source file and saves in a class file with a .class extension.


Text Editor
Source Code (filename.java)
Java Compiler
The Process to compile a java program...
bytecodes (classname.class)
Java Virtul Machine
Java Interpreter
Operating System

➢ **INTERPRETER**

Java interpreter translates the Java byte code into the code that can be understood by the Operating System. Basically, A Java interpreter is software that implements the Java virtual machine and runs Java applications. As the Java compiler compiles the source code into the Java byte code, the same way the Java interpreter translates the Java byte code into the code that can be understood by the Operating System.

➢ **DEBUGGER**

Java debugger helps in finding and the fixing of bugs in Java language programs. The Java debugger is denoted as jdb. It works like a command-line debugger for Java classes.

➢ **APPLET VIEWER**

Applet viewer is a command line program to run Java applets. It is included in the SDK. It helps you to test an applet before you run it in a browser. An applet is a special type of application that's included as a part of an HTML page and can be stored in a web page and run within a web browser. The applet's code gets transferred to the system and then the Java Virtual Machine (JVM) of the browser executes that code and displays the output.

JAVA IS SUITABLE FOR DESKTOP APPLICATION

There are several applications built on Java technology. Development tools like Eclipse is one the popular example among them. From enterprise applications, Lotus Notes is another good example of desktop Java applications. Such examples show that Java can be used extensively to develop desktop applications apart from installation run times and splash screens. Let us first examine what Java can offer and what are the benefits of using Java as development technology for stand alone applications.

- Independence of Java
- Security
- Manageability
- Pluggable User Interface
- Open Source
- Remote Access
- Networking Features

JAVA VIRTUAL MACHINE

The JVM is an important part of Java. The existence of JVMs on numerous different devices and software programs has allowed programs written in Java to be run on a large number of devices. You can write a single program with Java, and then that program can be run on numerous operating system.

The JVM must also deal with the conversion of type and the creation and control of objects. In addition to this, it will also need to deal with control transfers and throwing exceptions. The goal of the Java Virtual Machine is to make sure the code is compatible with the hardware or software that it is being converted into. For example, each OS will need to have an implentation for JVM that is unique. While the JVM will always study the byte code in the same way, the implementation of the code will likely be different. The most complicated aspect of this is the use of the Java API.

JSP TECHNOLOGY AND JAVA SERVLETS

JSP technology uses XML-like tags that encapsulate the logic that generates the content for the page. The application logic can reside in server-based resources (such as JavaBeans component architecture) that the page accesses with these tags. Any and all formatting (HTML or XML) tags are passed directly back to the response page.

y separating the page logic from its design and display and supporting a reusable component-based design, JSP technology makes it faster and easier than ever to build Web-based applications. Java Server Pages technology is an extension of the Java Servlet technology. Servlets are platform-independent, server-side modules that fit seamlessly into a Web server framework and can be used to extend the capabilities of a Web server with minimal overhead, maintenance, and support.

Unlike other scripting languages, servlets involve no platform-specific consideration or modifications; they are application components that are downloaded, on demand, to the part of the system that needs them. Together, JSP technology and servlets provide an attractive alternative to other types of dynamic Web scripting/programming by offering: platform independence; enhanced performance; separation of logic from display; ease of administration; extensibility into the enterprise; and, most importantly, ease of use. Today servlets are a popular choice for building interactive Web applications. Third-party servlet containers are available for Apache Web Server, Microsoft IIS, and others. Servlet containers are usually a component of Web and application servers.

DATA SET

A data set is a collection of data, usually presented in tabular form. Each column represents a particular variable. Each row corresponds to a given member of the data set in question. Its values for each of the variables, such as height and weight of an object or values of random numbers. Each value is known as a datum. The data set may comprise data for one or more members, corresponding to the number of rows. A data set has several characteristics which define its structure and properties. These include the number and types of the attributes or variables and the various stastical measures which may be applied to them such as standard deviation and kurtosis.

In the simplest case, there is only one variable, and then the data set consist of a single column of values, often represented as a list. In spite of the name, such a univariate data set is not a set in the usual mathematical sense, since a given value may occur multiple times. Normally the order does not matter, and then the collection of values may be considered to be a multiset rather than an (ordered) list.

DEFINING THE PROBLEM

EXISTING SYSTEM

In Existing, 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. Manual image annotation is time-consuming, laborious and expensive to address this; there has been a large amount of research done on automatic image annotation. Typically, in the development of an image requisition system, semantic image retrieval relies heavily on the related captions, e.g., file-names, categories, annotated keywords, and other manual descriptions. This kind of textual-based image retrieval always suffers from two problems: high-priced manual annotation and inappropriate automated annotation. On one hand, high-priced manual annotation cost is prohibitive in coping with a large-scale data set. On the other hand, inappropriate automated annotation yields the distorted results for semantic image retrieval. Relevance feedback-based CBIR methods usually request a number of iterative feedbacks to produce refined search results, especially in a large-scale image database. This is impractical and inefficient in real applications.

PROBLEM IN EXISTING SYSTEM

❖ Problem of image annotation
- Large volumes of databases
- Valid only for one language – with image retrieval this limitation should not exist.

❖ Problem of human perception
- Subjectivity of human perception.
- Too much responsibility on the end-user.

❖ Problem of deeper (abstract) needs
- Queries that cannot be described at all, but tap into the visual features of images.

PROPOSED SYSTEM

We proposed to growing interest in CBIR because of the limitations inherent in metadata-based systems, as well as the large range of possible uses for efficient image retrieval. Textual information about images can be easily searched using existing technology, but requires humans to personally describe every image in the database. This is impractical for very large databases, or for images that are generated automatically, e.g. from surveillance cameras. It is also possible to miss images that use different synonyms in their descriptions. Systems based on categorizing images in semantic classes like "cat" as a subclass of "animal" avoid this problem but still face the same scaling issues. CBIR presents an image conceptually, with a set of low-level visual features such as color, texture, and shape. These conventional approaches for image retrieval are based on the computation of the similarity between the user's query and images via a query. Navigation-Pattern-based Relevance Feedback (NPRF), to achieve the high efficiency and effectiveness of CBIR in coping with the large-scale image data.

In terms of efficiency, the iterations of feedback are reduced substantially by using the navigation patterns discovered from the user query log. In terms of effectiveness, our proposed search algorithm NPRF Search makes use of the discovered navigation patterns and three kinds of query refinement strategies, Query Point Movement (QPM), Query Reweighing (QR), and Query Expansion (QEX), to converge the search space toward the user's intention effectively. By using NPRF method, high quality of image retrieval on RF can be achieved in a small number of feedbacks.

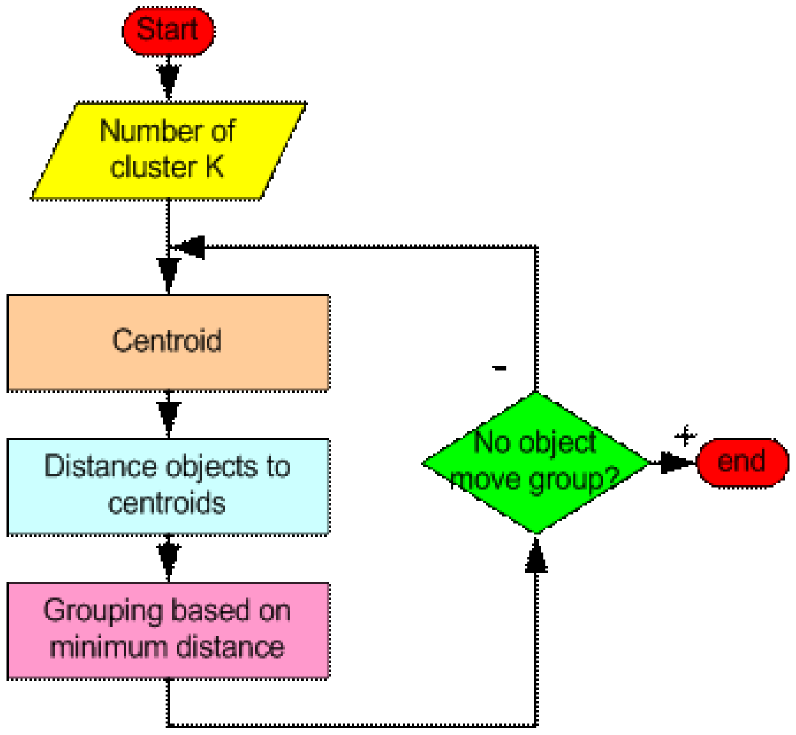
K-MEANS ALGORITHM

K-Means Clustering is a method of cluster analysis which aims to partition observations into *k* clusters in which each observation belongs to the cluster with the nearest mean. It is similar to the expectation-maximization algorithm for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data as well as in the iterative refinement approach.

$$\sum_{i \in \text{clusters}} \left\{ \sum_{j \in \text{elements of i' th cluster}} \left\| x_j - \mu_i \right\|^2 \right\}$$

These three steps are comes under K-Mean algorithm:
- Shape Features Extraction
- Pair wise Shape Matching
- Meta Similarity

K-MEANS


Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
end

Improved algorithm needs significantly less iteration. Experimental results shows that choosing initial centroids by our algorithm is stable when compared with randomly selected Initial centroids.

TWO COMPUTATIONAL APPROACHES

SPECTRAL MATCHING

Combines LSDs and Global shape constraints using pair wise geometric similarities.

META SIMILARITY

Agglomerates pair wise shape similarity to reject outliers and improve the retrieval accuracy.

CONTENT BASED RETRIEVAL

Content-based image retrieval systems work with whole images and searching is based on comparison of the query. The commonest features used are mathematical measures of color, texture or shape. The system then identifies those stored images whose feature values match those of the query most closely and displays thumbnails of these images on the screen.

COLOR

Color is a property that depends on the reflection of light to the eye and the processing of that information in the brain. We use color everyday to tell the difference between objects, places, and the time of day. Usually colors are defined in three dimensional color spaces. These could either be **RGB** (Red, Green, and Blue), **HSV** (Hue, Saturation, and Value) or **HSB** (Hue, Saturation, and Brightness).

The last two are dependent on the human perception of hue, saturation, and brightness. Most image formats such as **JPEG, BMP, GIF**, use the RGB color space to store information. The RGB color space is defined as a unit cube with red, green, and blue axes.

TEXTURE

Texture is that innate property of all surfaces that describes visual patterns, each having properties of homogeneity. It contains important information about the structural arrangement of the surface, such as clouds, leaves, bricks, fabric, etc. It also describes the relationship of the surface to the surrounding environment. In short, it is a feature that describes the distinctive physical composition of a surface.

SHAPE

Shape may be defined as the characteristic surface configuration of an object; an outline or contour. It permits an object to be distinguished from its surroundings by its outline.

Shape representations can be generally divided into two categories

- Boundary-based and
- Region-based.

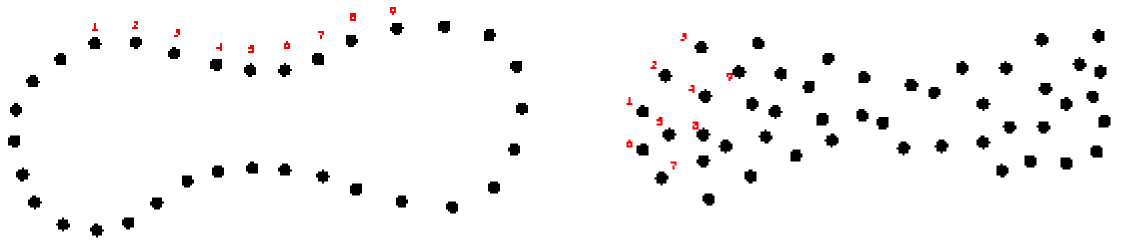


Figure: Boundary-based & Region-based

Boundary-based shape representation only uses the outer boundary of the shape. This is done by describing the considered region using its external characteristics; i.e., the pixels along the object boundary. Region-based shape representation uses the entire shape region by describing the considered region using its internal characteristics; i.e., the pixels contained in that region.

III. LITERATURE REVIEW

COLOR CONTENT-BASED IMAGE CLASSIFICATION

In content-based image retrieval systems the most efficient and simple searches are the color-based searches, which can be realized in several color spaces and by several color descriptors. In this paper the possibility of image classification using certain color descriptors is examined, and the usage of different color spaces and descriptors depending on the image database domain is presented.

The color has great importance in content-based image retrieval systems, which is stored in the intensity vectors of image pixels, and this information can be retrieved simply. However the color can be represented in different color spaces and that is not irrelevant which color space is used in an application. The color information of an image represented in an arbitrary color space can be stored on several ways, but that is strongly application dependent which representation method is the most efficient at a determined search.

LOCALIZED CONTENT-BASED IMAGE RETRIEVAL

We define localized content-based image retrieval as a CBIR task where the user is only interested in a portion of the image, and the rest of the image is irrelevant. In this paper we present a localized CBIR system, ACCIO!, that uses labeled images in conjunction with a multiple-instance learning algorithm to first identify the desired object and weight the features accordingly,

and then to rank images in the database using a similarity measure that is based upon only the relevant portions of the image. A challenge for localized CBIR is how to represent the image to capture the content.

We present and compare two novel image representations, which extend traditional segmentation based and salient point-based techniques respectively, to capture content in a localized CBIR setting.

CONTENT-BASED IMAGE RETRIEVAL FOR DIGITAL FORENSICS

Digital forensic investigators are often faced with the task of manually examining a large number of (photographic) images in order to identify potential evidence. The task can be especially daunting and time-consuming if the target of the investigation is very broad, such as a web hosting service. Current forensic tools are woefully inadequate in facilitating this process and are largely confined to generating pages of thumbnail images and identifying known files through cryptographic hashes.

We present a new approach that significantly automates the examination process by relying on image analysis techniques. The general approach is to use previously identified content (e.g., contraband images) and to perform feature extraction, which captures mathematically the essential properties of the images. Based on this analysis, we build a feature set database that allows us to automatically scan a target machine for images that are similar to the ones in the database.

An important property of our approach is that it is not possible to recover the original image from the feature set. Therefore, it becomes possible to build a (potentially very large) database targeting known contraband images that investigators may be barred from collecting directly.

The same approach can be used to automatically search for case-specific images, contraband or otherwise, or to provide online monitoring of shared storage for early detection of certain images.

LOCAL VS GLOBAL HISTOGRAM-BASED IMAGE CLUSTERING

We present two image clustering techniques to automatically group color images that correlate with semantic concepts. This work goes towards satisfying the ever growing need for techniques that are capable of automatically generating semantic concepts for images from their visual features. We present two techniques and evaluate their relative performances based on the perceptual similarity among images that are clustered together. The first technique is based on the localized histogram information while the second approach uses global histogram information to characterize the images. Experiments using a 2100 image database are presented to show the relative effectiveness of the presented systems. Preliminary results show that the local histogram approach gives better clustering results and its characterization of images is more closely related to the way in which humans perceive images.

IV. SYSTEM ANALYSIS

HARDWARE SPECIFICATION

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design.

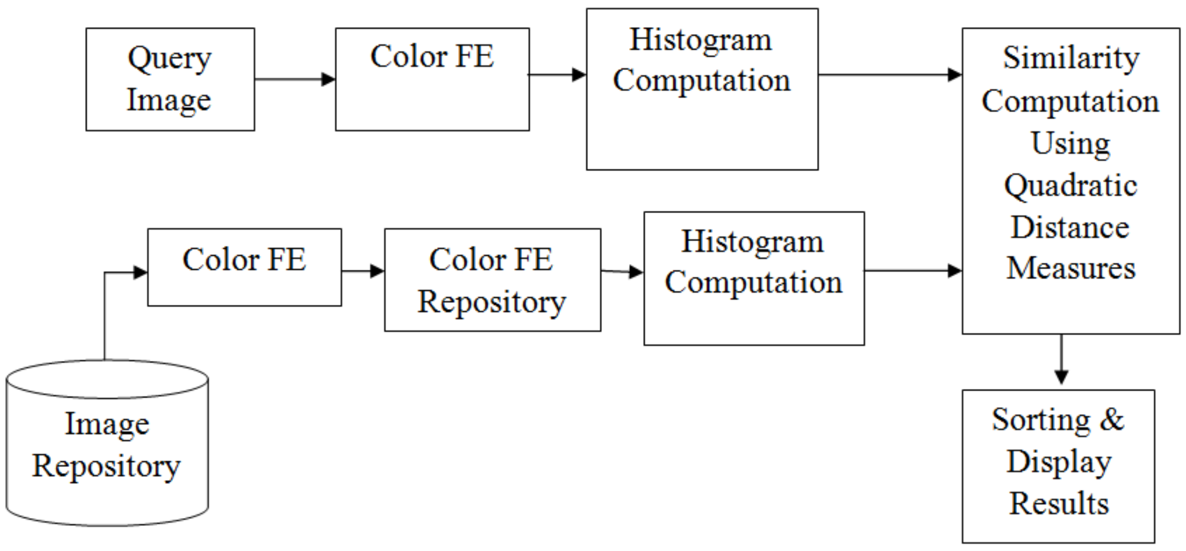
System	: Pentium4 3.5GHz
Hard Disk	: 40 GB
Floppy Drive	: 1.44MB
Monitor	: 15 VGA colour
Mouse	: Logitech.
RAM	: 1 GB
Keyboard	: 110 keys enhanced

SOFTWARE SPECIFICATION

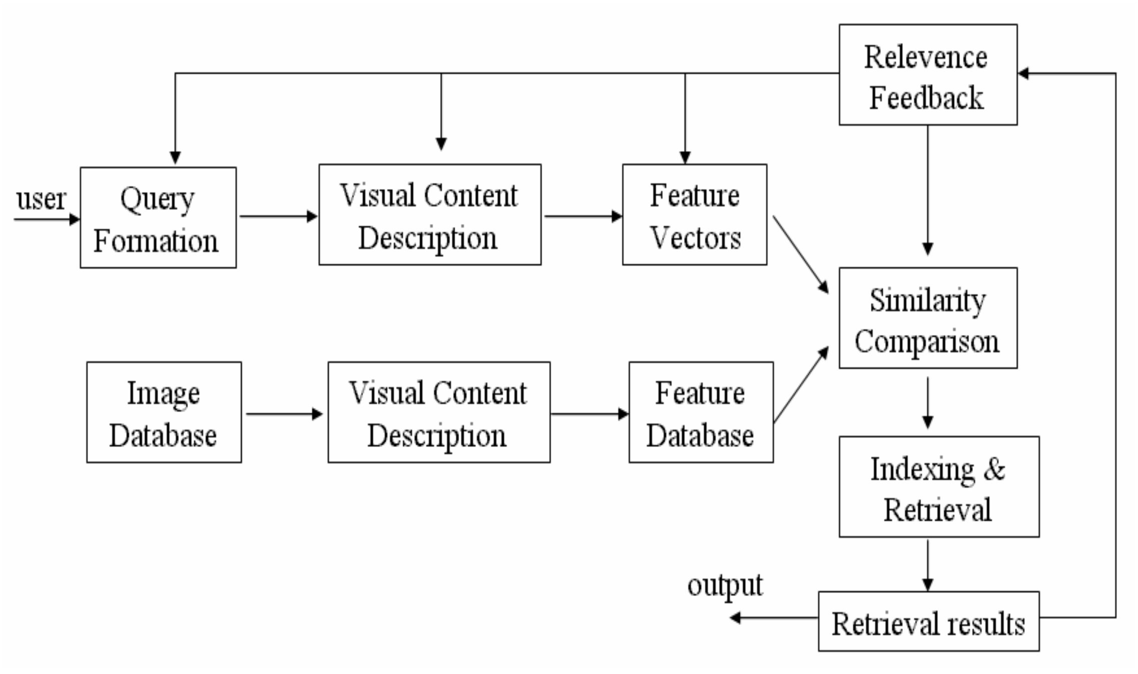
The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

Operating system	: Windows XP Professional
Framework	: Java
Front End	: Jsp, Servlet
Coding Language	: Java
Back End	: Data Set

SYSTEM DESIGN
MODULE DIAGRAM



OBJECT DIAGRAM



V. FEASIBILITY STUDY

INTRODUCTION

Feasibility study is the test of a system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of recourses. It focuses on the evaluation of existing system and procedures analysis of alternative candidate system cost estimates. Feasibility analysis was done to determine whether the system would be feasible.

The development of a computer based system or a product is more likely plagued by resources and delivery dates. Feasibility study helps the analyst to decide whether or not to proceed, amend, postpone or cancel the project, particularly important when the project is large, complex and costly. Once the analysis of the user requirement is complement, the system has to check for the compatibility and feasibility of the software package that is aimed at. An important outcome of the preliminary investigation is the determination that the system requested is feasible.

TECHNICAL FEASIBILITY

The technology used can be developed with the current equipments and has the technical capacity to hold the data required by the new system.

- This technology supports the modern trends of technology.
- Easily accessible, more secure technologies.

Technical feasibility on the existing system and to what extend it can support the proposed addition. We can add new modules easily without affecting the Core Program. Most of parts are running in the server using the concept of stored procedures.

OPERATIONAL FEASIBILITY

This proposed system can easily implement, as this is based on PHP and Angular JS. The database created is with MySQL Server which is more secure and easy to handle. The resources that are required to implement/install these are available. The personal of the organization already has enough exposure to computers. So the project is operationally feasible.

ECONOMICAL FEASIBILITY

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. An Entrepreneur must accurately weigh the cost versus benefits before taking an action. This system is more economically feasible which assess the brain capacity with quick & online test. So it is economically a good project.

VI. SYSTEM TESTING

INTRODUCTION

Software testing is an important element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of S/W as a system element and costs associated with software failure are motivating forces for well planned, through testing.

Though the test phase is often thought of as separate and distinct from the development effort—first develops, and then test—testing is a concurrent process that provides valuable information for the development team.

There are at least three options for integrating Project Builder into the test phase

- ❖ Testers do not install Project Builder, use Project Builder functionality to compile and source-control the modules to be tested and hand them off to the testers, whose process remains unchanged.
- ❖ The testers import the same project or projects that the developers use.
- ❖ Create a project based on the development project but customized for the testers (for example, it does not include support documents, specs, or source), who import it.

TESTING OBJECTIVES

There are several rules that can serve as testing objectives

They are,

- ❖ 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 undiscovered error.
- ❖ A successful test is one that uncovers an undiscovered error.

If testing is conducted successfully according to the objectives stated above, it will uncover errors in the software.

TESTING TYPES

- ➢ Black –Box Testing
- ➢ White-Box Testing
- ➢ Unit Testing
- ➢ Integration Testing
- ➢ Validation Testing

BLACK-BOX TESTING

It is focused on the functional requirements of the software. It is not an alternative to White Box Testing; rather, it is a complementary approach that is likely to uncover a different class of errors than White Box methods. It is attempted to find errors in the following categories.

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access.

WHITE-BOX TESTING

This test is conducted during the code generation phase itself. All the errors were rectified at the moment of its discovery. During this testing, it is ensured that

- All independent paths within a module have been exercised at least one.

UNIT TESTING

This is the first level of testing. In this different modules are tested against the specifications produced during the design of the module. During this testing the number of arguments is compared to input parameters, matching of parameter etc. It is also ensured whether the file attributes are correct, whether the files are opened before using, whether input/output errors are handled etc. Unit test is conducted using a test Driver usually.

INTEGRATION TESTING

Integration Testing is a Systematic Testing for constructing the program structure, while at the same time conducting test to uncover errors associated within the interface. Bottom-up integration is used for this phase. It begins construction and testing with atomic modules. This strategy is implemented with the following steps.

VALIDATION TESTING

This provides the final assurance that the software meets all functional, behavioral and performance requirements. This software is completely assembled as a package.

OUTPUT TESTING

- Output of test cases compared with the expected results created during design of test cases.

- Asking the user about the format required by them tests the output generated or displayed by the system under consideration.
- Here, the output format is considered into two was, one is on screen and another one is printed format.
- The output on the screen is found to be correct as the format was designed in the system design phase according to user needs.
- The output comes out as the specified requirements as the user’s hard copy.

VII. APPLICATIONS

- **Crime prevention:** Automatic face recognition systems, used by police forces.
- **Security Check:** Finger print or retina scanning for access privileges.
- **Medical Diagnosis:** Using CBIR in a medical database of medical images to aid diagnosis by identifying similar past cases.
- **Intellectual Property:** Trademark image registration, where a new candidate mark is compared with existing marks to ensure no risk of confusing property ownership.

VIII. CONCLUSION

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

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

FUTURE ENHANCEMENT

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

BIBILIOGRAPGY

[1]. Y. Zhao and D. Cheng, ‘‘On controllability and stabilizability of probabilistic Boolean control networks,’’ Sci. China Inf. Sci., vol. 57, no. 1, pp. 1–14, 2014

[2]. H. Li, L. Xie, and Y. Wang, ‘‘On robust control invariance of Boolean control networks,’’ Automatica, vol. 68, pp. 392–396, Jun. 2016

[3]. L. Tong, Y. Liu, F. E. Alsaadi, and T. Hayat, ‘‘Robust sampled-data control invariance for Boolean control networks,’’ J. Franklin Inst., vol. 354, pp. 7077–7087, Oct. 2017.

[4]. R. Pal, A. Datta, and E. R. Dougherty, ‘‘Robust intervention in probabilistic Boolean networks,’’ IEEE Trans. Signal Process., vol. 56, no. 3, pp. 1280–1294, Mar. 2008

[5]. F. Li, J. Li, and Z. Yu, ‘‘Robust control invariance of probabilistic Boolean control networks,’’ in Proc. IEEE Conf. IECON, Oct. 2016, pp. 5398–5402.

[6]. F. Li and T. Yang, ‘‘Robust reachability of Boolean control networks,’’ IEEE/ACM Trans. Comput. Biol. Bioinf., vol. 14, no. 3, pp. 740–745, May 2017.

[7]. Y. Zhao and D. Cheng, ‘‘On controllability and stabilizability of probabilistic Boolean control networks,’’ Sci. China Inf. Sci., vol. 57, no. 1, pp. 1–14, 2014.

[8]. F. Li and Y. Tang, ‘‘Set stabilization for switched Boolean control networks,’’ Automatica, vol.

[9].] D. Cheng, H. Qi, and Z. Li, Analysis and Control of Boolean Networks: A Semi-Tensor Product Approach. London, U.K.: Springer-Verlag, 2011.

[10]. Z. Li and H. Xiao, ‘‘Weak reachability of probabilistic Boolean control networks,’’ in Proc. Int. Conf. Adv. Mech. Syst., Aug. 2015, pp. 56–60.

APPENDIX – A

```
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.image.*;
import javax.imageio.*;
import java.lang.*;
import java.text.*;
import beans.*;
public class  cbir
{
Vector<ImageData> dataSet;
public cbir(String path){
try
{
dataSet = new Vector<ImageData>();
BufferedReader br = new BufferedReader(new FileReader( new File(path, "data.txt")));
while(true)
{
ImageData imData = new ImageData();

String line = br.readLine();
if (line == null) break;
if (line.startsWith("#")) continue;
StringTokenizer st = new StringTokenizer(line);
if (st.countTokens() < 2) continue;
imData.setImageName(st.nextToken());
imData.setImageClass(st.nextToken());
dataSet.add(imData);
}
br.close();

for (int i=0;i<dataSet.size();i++ )
{
ImageData imData = (ImageData)dataSet.get(i);

BufferedImage bi = ImageIO.read(new File(path,imData.getImageName()));
CBIRImage cbirImage = new CBIRImage(bi);
Vector histogram = Features.getGlobalColorHistogram(cbirImage);
Vector<Double> statFeature = new Vector<Double>();

double mean = Features.getMean(histogram);
double sd = Features.getSD(histogram, mean);
double skew = Features.getSkew(histogram, mean, sd);
double energy = Features.getEnergy(histogram);
double entropy = Features.getEntropy(histogram);

statFeature.add(new Double(mean));
statFeature.add(new Double(sd));
statFeature.add(new Double(skew));
statFeature.add(new Double(energy));
statFeature.add(new Double(entropy));

imData.setImageFeature(histogram);
imData.setImageStatFeature(statFeature);
imData.setImageMean(mean);
imData.setImageSD(sd);
imData.setImageSkew(skew);
imData.setImageEnergy(energy);
imData.setImageEntropy(entropy);

}
}
catch (Exception e){e.printStackTrace();}
```

```
}

public Vector getDataSet() { return this.dataSet; }
}

import java.awt.geom.AffineTransform;
import java.awt.image.*;
import java.io.*;
import javax.imageio.ImageIO;

public class CBIRImage
{
public CBIRImage(String filename)
{
_image = null;
_filename = filename;
}
public CBIRImage(File fileHandle)
{
_image = null;
try
{
_filename = fileHandle.getCanonicalPath();
}
catch(IOException e)
{
e.printStackTrace();
}
}
public CBIRImage(BufferedImage image)
{
_image = image;
_filename = "temp";
}
public CBIRImage(BufferedImage image, String filename)
{
_image = image;
_filename = filename;
}
public boolean readImage()
{
try
{
_image = ImageIO.read(new File(_filename));
_colorModel = _image.getColorModel();
}
catch(Exception e)
{
System.out.println("read Image="+_filename);
return false;
}
return true;
}
public int getWidth()
{
return _image.getWidth();
}
public int getHeight()
{
return _image.getHeight();
```

```
}
public String getFileName()
{
return _filename;
}
public int[] getPackedRGBPixel()
{
return _image.getRGB(0, 0, _image.getWidth(), _image.getHeight(), null, 0, _image.getWidth());
}
public int[] getRGBPixel()
{
int size = _image.getHeight() * _image.getWidth();
int pixel[] = new int[size];
int pixels[] = new int[size * 3];
PixelGrabber pg = new PixelGrabber(_image, 0, 0, _image.getWidth(), _image.getHeight(), pixel, 0,
_image.getWidth());
try
{
pg.grabPixels();
}
catch(InterruptedException e)
{
e.printStackTrace();
return null;
}
if((pg.getStatus() & 0x80) != 0)
{
System.out.println("Error in grabbing pixel");
return null;
}
for(int i = 0; i < size; i++)
{
pixels[i * 3] = pixel[i] >> 16 & 0xff;
pixels[i * 3 + 1] = pixel[i] >> 8 & 0xff;
pixels[i * 3 + 2] = pixel[i] & 0xff;
}
return pixels;
}
public int getPackedRGBPixel(int x, int y)
{
return _image.getRGB(x, y);
}
public int[] getRGBPixel(int x, int y)
{
int pixel[] = new int[1];
int pixels[] = new int[3];
PixelGrabber pg = new PixelGrabber(_image, x, y, 1, 1, pixel, 0, _image.getWidth());
try
{
pg.grabPixels();
}
catch(InterruptedException e)
{
e.printStackTrace();
return null;
}
if((pg.getStatus() & 0x80) != 0)
{
System.out.println("Error in grabbing pixel");
return null;
} else
{
pixels[0] = pixel[0] >> 16 & 0xff;
pixels[1] = pixel[0] >> 8 & 0xff;
pixels[2] = pixel[0] & 0xff;
return pixels;
}
}

public boolean generateThumbnail(String filename, int width, int height)
{
int originalWidth = _image.getWidth();
int originalHeight = _image.getHeight();
double x = (double)width / (double)originalWidth;
double y = (double)height / (double)originalHeight;
AffineTransformOp transformFilter = new AffineTransformOp(new AffineTransform(x, 0.0D, 0.0D, y, 0.0D,
0.0D), 1);
BufferedImage dest = new BufferedImage(width, height, 5);
transformFilter.filter(_image, dest);
File output = new File(filename);
try
{
ImageIO.write(dest, "jpg", output);
}
catch(IOException e)
{
return false;
}
return true;
}

public boolean generateDefaultThumbnail(String filename)
{
int x;
int y;
if(_image.getWidth() > _image.getHeight())
{
x = 100;
y = (_image.getHeight() * 100) / _image.getWidth();
} else
{
y = 100;
x = (_image.getWidth() * 100) / _image.getHeight();
}
return generateThumbnail(filename, x, y);
}
public void resize(int width, int height)
{
int originalWidth = _image.getWidth();
int originalHeight = _image.getHeight();
double x = (double)width / (double)originalWidth;
double y = (double)height / (double)originalHeight;
AffineTransformOp transformFilter = new AffineTransformOp(new AffineTransform(x, 0.0D, 0.0D, y, 0.0D,
0.0D), 2);
BufferedImage dest = new BufferedImage(width, height, _image.getType());
transformFilter.filter(_image, dest);
_image = null;
_image = dest;
}
```

```
public CBIRImage getResizedImage(int width, int height)
{
int originalWidth = _image.getWidth();
int originalHeight = _image.getHeight();
double x = (double)width / (double)originalWidth;
double y = (double)height / (double)originalHeight;
AffineTransformOp transformFilter = new AffineTransformOp(new AffineTransform(x, 0.0D, 0.0D, y, 0.0D,
0.0D), 2);
BufferedImage dest = new BufferedImage(width, height, _image.getType());
transformFilter.filter(_image, dest);
return new CBIRImage(dest, _filename);
}

public void blur()
{
float weight = 0.1111111F;
float elements[] = new float[9];
for(int i = 0; i < 9; i++)
elements[i] = weight;

Kernel myKernel = new Kernel(3, 3, elements);
ConvolveOp simpleBlur = new ConvolveOp(myKernel, 1, null);
if(_colorModel instanceof IndexColorModel)
{
BufferedImage des = new BufferedImage(_image.getWidth(), _image.getHeight(), 5);
simpleBlur.filter(_image, des);
_image = des;
} else
{
BufferedImage des = new BufferedImage(_image.getWidth(), _image.getHeight(), _image.getType());
simpleBlur.filter(_image, des);
_image = des;
}
}

protected void finalize()
throws Throwable
{
_image = null;
_filename = null;
_colorModel = null;
}

private BufferedImage _image;
private String _filename;
private ColorModel _colorModel;
}
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.image.*;
import javax.imageio.*;
import java.lang.*;
import java.text.*;
import beans.*;
public abstract class  Classifier
{
public static Vector classify(ImageData input, Vector dataSet)
{
Vector<Vector> result = new Vector<Vector>();
```

```
LinkedHashMap<String, Double> imMap = new  LinkedHashMap<String, Double>();
LinkedHashMap<String, Double> imStatMap = new  LinkedHashMap<String, Double>();

for (int x=0;x< dataSet.size();x++ )
{
double distance =
getDistanceMeasure(input.getImageFeature(),((ImageData)dataSet.get(x)).getImageFeature());
imMap.put(((ImageData)dataSet.get(x)).getImageName(),new Double(distance));
double statDistance =
getDistanceMeasure(input.getImageStatFeature(),((ImageData)dataSet.get(x)).getImageStatFeature());
imStatMap.put(((ImageData)dataSet.get(x)).getImageName(),new Double(statDistance));
}
LinkedHashMap imSortedMap = getSorted(imMap);
LinkedHashMap imStatSortedMap = getSorted(imStatMap);
Vector r1 = getImageData(imSortedMap, dataSet);
Vector r2 = getImageData(imStatSortedMap, dataSet);
result.add(r1);
result.add(r2);
return result;
}

public static double getDistanceMeasure(Vector a, Vector b)
{
int noOfDimension = a.size();
if(noOfDimension != b.size())
return 99999.899999999994D;
double total = 0.0D;
for(int i = 0; i < noOfDimension; i++)
{
double doubleA = ((Double)a.elementAt(i)).doubleValue();
double doubleB = ((Double)b.elementAt(i)).doubleValue();
total += (doubleA - doubleB) * (doubleA - doubleB);
}

return Math.sqrt(total / (double)noOfDimension);
}
public static LinkedHashMap getSorted(LinkedHashMap imMap) {

LinkedHashMap<String, Double> imSortedMap = new  LinkedHashMap<String, Double>();

java.util.List imMapKeys = new ArrayList(imMap.keySet());
java.util.List imMapValues = new ArrayList(imMap.values());
TreeSet sortedSet = new TreeSet(imMapValues);
Object[] sortedArray = sortedSet.toArray();
int size = sortedArray.length;

for (int i=0; i<size; i++) {
imSortedMap.put((String)imMapKeys.get(imMapValues.indexOf(sortedArray[i])),((Double)sortedArray[i]).do
ubleValue());
}
return imSortedMap;
}
public static Vector getImageData(LinkedHashMap imageMap, Vector dataSet) {

Vector<ImageData> result = new Vector<ImageData>(10);
Iterator iterator = imageMap.keySet().iterator();
int c = 0;
while( iterator. hasNext() ){
String key = (String) iterator.next();
for (int x=0;x< dataSet.size();x++ ){
```

```
ImageData imData = (ImageData)dataSet.get(x);
if (imData.getImageName().equals(key)){
imData.setImageDistance(((Double)imageMap.get(key)).doubleValue());
result.add(imData);
}
}
c++;
if (c == 10) break;
}
return result;
}
}
import java.io.*;
import java.util.*;

public abstract class Features
{
public static Vector getGlobalColorHistogram(CBIRImage image)
{
Vector globalColorHistogram = new Vector(64);
int width = image.getWidth();
int height = image.getHeight();
int size = width * height;
int pixels[] = image.getPackedRGBPixel();
double histogram[] = new double[64];
for(int i = 0; i < 64; i++)
histogram[i] = 0.0D;
for(int i = 0; i < size; i++)
{
int k = pixels[i] >> 18 & 0x30 | pixels[i] >> 12 & 0xc | pixels[i] >> 6 & 3;
histogram[k]++;
}
for(int i = 0; i < 64; i++)
globalColorHistogram.add(new Double(histogram[i] / (double)size));
return globalColorHistogram;   }
public static double getMean(Vector histogram){
double mean = 0.0D;
double sum = 0.0D;
for (int i=0;i<histogram.size() ;i++ )
{
sum += i*((Double)histogram.get(i)).doubleValue();
}
return (double)sum;
}
public static double getSD(Vector histogram, double mean){
double sum = 0.0D;
for (int i=0;i<histogram.size() ;i++ )
{
sum += Math.pow((i- mean),2)*((Double)histogram.get(i)).doubleValue();
}
return Math.sqrt((double)sum);
}
public static double getSkew(Vector histogram, double mean, double sd) {
double sum = 0.0D;
for (int i=0;i<histogram.size() ;i++ ) {
sum += Math.pow((i- mean),3)*((Double)histogram.get(i)).doubleValue();
}
return (double)sum/Math.pow(sd,3);
}
public static double getEnergy(Vector histogram) {
```

```
double energy = 0.0D;
for (int i=0;i<histogram.size() ;i++ )
{
energy += Math.pow(((Double)histogram.get(i)).doubleValue(),2);
}
return energy;
}
public static double getEntropy(Vector histogram) {
double entropy = 0.0D;
for (int i=0;i<histogram.size() ;i++ )
{
double  pg = ((Double)histogram.get(i)).doubleValue();
if (pg != 0.0D)     {
entropy += pg*Math.log(pg) ;
}
}
return -1*entropy;
}
}
import java.io.*;
import java.util.*;
import beans.*;
import java.awt.*;
import java.awt.image.*;
import javax.imageio.*;
import javax.servlet.*;
import javax.servlet.http.*;
import org.apache.commons.fileupload.DiskFileUpload;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.FileItemFactory;
import org.apache.commons.fileupload.FileUpload;
import org.apache.commons.fileupload.FileUploadException;

public class QueryServlet extends HttpServlet
{
cbir cb;
public void init(ServletConfig config) throws ServletException {
super.init(config);
cb = new cbir(getServletContext().getRealPath("DataSet/"));
}
public void doPost(HttpServletRequest req,HttpServletResponse res) throws ServletException,IOException
{
String queryContentType= "";
File queryImage = null;
String queryImageName = "";
CBIRImage cbirImage;

try
{
System.out.println(getServletContext().getRealPath("DataSet/"));
boolean isMultipart = FileUpload.isMultipartContent(req);
DiskFileUpload upload = new DiskFileUpload();
java.util.List items = upload.parseRequest(req);
Iterator iter = items.iterator();
while (iter.hasNext())
{
FileItem item = (FileItem) iter.next();
if (!item.isFormField()) {
queryContentType = item.getContentType();
queryImageName = new File(item.getName()).getName();
queryImage=new File(getServletContext().getRealPath("QueryFiles/"),queryImageName);
item.write(queryImage);
}
}
Thread.sleep(5000);
BufferedImage bi = ImageIO.read(queryImage);
cbirImage = new CBIRImage(bi);
Vector histogram = Features.getGlobalColorHistogram(cbirImage);
Vector<Double> statFeature = new Vector<Double>();
double mean = Features.getMean(histogram);
double sd = Features.getSD(histogram, mean);
double skew = Features.getSkew(histogram, mean, sd);
double energy = Features.getEnergy(histogram);
double entropy = Features.getEntropy(histogram);

statFeature.add(new Double(mean));
statFeature.add(new Double(sd));
statFeature.add(new Double(skew));
statFeature.add(new Double(energy));
statFeature.add(new Double(entropy));
ImageData imData = new ImageData();
imData.setImageName(queryImage.getName());
imData.setImageType(queryContentType);
imData.setImageFeature(histogram);
imData.setImageStatFeature(statFeature);
imData.setImageMean(mean);
imData.setImageSD(sd);
imData.setImageSkew(skew);
imData.setImageEnergy(energy);
imData.setImageEntropy(entropy);
Vector<ImageData> dataSet = cb.getDataSet();
Vector result = Classifier.classify(imData, dataSet);
Vector m1 = (Vector)result.get(0);
Vector m2 = (Vector)result.get(1);


req.setAttribute("m1", m1);
req.setAttribute("m2", m2);
req.setAttribute("qmData", imData);
getServletConfig().getServletContext().getRequestDispatcher("/home.jsp").forward(req, res);
}
catch(Exception e)
{
e.printStackTrace();
req.setAttribute("error", e.getMessage());
        getServletConfig().getServletContext().getRequestDispatcher("/home.jsp").forward(req, res);
}
}
}
```