Elimination of Repeated Occurrences in Image Search Engines

Saed Alqaraleh

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
January 2011
Gazimagusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director (a)

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Assoc. Prof. Dr. Muhammed Salamah
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Assoc.Prof.Dr. Işık AYBAY
Supervisor

Examining Committee

1. Assoc.Prof.Dr. Işık AYBAY
2. Assoc. Prof. Dr. Muhammed Salamah
3. Asst. Prof. Dr. Gürcü ÖZ

ABSTRACT

We propose a new method for elimination of repeated occurrences in image search engines. We have built software that: Compares images in a database, and marks only one copy of repeating files using a hashing technique. Marking one of the repeating images will lead to faster access and will eliminate the repetition of the same images more than once. The software can work periodically, for dealing with any updates on the image database.

We have developed another version of the software to be multipurpose, making use of the query by example tool, and it can also find images which are similar to each other within some percentages limits.

Keywords: Image Search Engines, Query by Example, Hash Algorithm, Information Retrieval.

ÖZ

Resim arama motorlarındaki tekrarlanan bulguları gidermek için yeni bir yöntem öneriyoruz. Geliştirdiğimiz yazılım: Veritabanındaki resimleri karşılaştırıyor, ve Hesaba dayalı adresleme (Hashing) tekniğini kullanarak tekrarlanan dosyaların bir kopyasını işaretliyor. Tekrarlanan resimlerin birini işaretlemek, daha hızlı erişim sağlıyor ve aynı resmin birden fazla görüntülenmesini engelliyor. Resim veritabanındaki güncellemelerle başa çıkmak için, yazılım periodik olarak çalıştırılabiliyor.

Örnek ile çalışan sorgu aracını kullanarak yazılımın bir diğer çok amaçlı versiyonu da geliştirilmiştir. Bu versiyonda yazılım benzer resimleri bazı yüzdelik sınırları kullanarak bulabiliyor.

Anahtar Kelimeler: Resim Arama Motorları, Örnek ile çalışan sorgu aracı, Hesaba Dayalı Adresleme Algoritması, Bilgi Erişimi.

DEDICATION

To My Family

(Especially to my Grandfather and my Grandmother Peace on their souls)

ACKNOWLEDGMENT

I would like to thank Assoc. Prof. Dr. Işık AYBAY for his guidance and continuous support through my study. Without his appreciated supervision, I would not be in this position.

I owe a big thank to my family. Thanks to my parents for their support through the period of my study. I will never forget my wife’s support, as she was beside me, and encouraging me all the time.

I would like to great my friends who were always around to support.

I know that saying thanks comparing with what they all have done is nothing. But all of them will be always in my Heart.

TABLE OF CONTENTS

ABSTRACT iii

ÖZ iv

DEDICATION v

ACKNOWLEDGMENT vi

1 INTRODUCTION 1

2 RELATED WORKS 4

2.1 Overview of Internet Search Engines 4

2.2 Overview of Related Work 5

2.2.1 Studies on Current Search Engine Mechanisms for Finding Images 5

2.2.2 Studies for Improving the Efficiency of Search Engines 9

2.2.2.1 Flexible and Extensible Framework for Web Image Retrieval 9

2.2.2.2 Direct Searching of Video Content (DIVAS) 10

2.2.2.3 SCENIQUE 10

2.2.2.4 Lazy 11

2.2.2.5 Query by Example 11

2.2.2.6 Query by Sketch 12

2.2.2.7 Hybrid Methods 12

2.2.2.8 Automatic Ranking of Websites 13

2.2.2.9 Key Block 13

2.2.2.10 Document Clustering 14

3 ELIMINATION OF REPEATED OCCURRENCES IN IMAGE SEARCHING 17

3.1 Programming Environment 17

3.2 The Database 17

3.3 Software Mechanism 18

3.3.1 Creating the Images Database 18

3.3.2 Computing the Hash Value 19

3.3.3 Comparing the Hash Value 20

3.4 User Interface 21

4 PERFORMANCE STUDIES 25

4.1 Introduction 25

4.2 Bit-Wise Comparisons 25

4.2.1 Sequential Execution 26

4.2.2 Parallel execution: Client - Server Architecture 27

4.3 Hash Comparison 31

4.4 Comparison of Hash Algorithm and Bit-wise techniques 33

4.5 Parallel Work with Hash Algorithm 34

4.6 Saving the Hash Values in the Database 36

4.7 Mechanism of Dividing the Work between Parallel Copies 36

4.8 Comparing dynamically way versus. Saving the Hash Values earlier in the Database 41

5 STUDIES ON FINDING SIMILAR IMAGES 43

5.1 Introduction 43

5.2 Query by Example Mechanism 43

5.3 Methodology Developed For Implementing the Query by Example Techniques 47

5.3.1 Bit- Wise Comparison 47

5.3.2 Exhaustive Template Matching 48

5.3.3 Comparison between Exhaustive Template Matching and Bit- Wise Comparison Techniques 50

CONCLUSION 52

APPENDICES 59

Appendix A: The source code of the module. 59

LIST OF TABLES

Table 2.1.Number of Images For Some Queries (Reachable By Google)..... 5

Table 4.1.Results of Sequential Comparison / Deletion for Base Image..... 26

Table 4.2.Results of Sequential Comparison / Deletion for Random Image. 27

Table 4.3.Results of the Client – Server Method for Base Images. 28

Table 4.4.Result of the Client –Server Method for Random Images. 30

Table 4.5.Comparison of SHA and MD5. 32

Table 4.6.Execution Times for Different Hash Algorithms..... 32

Table 4.7.Execution Time of Hash Algorithms and Bit Wise Comparison Technique.. 33

Table 4.8.Execution Time for 4 and 8 Clients. 34

Table 4.9.Execution Time Versus Number of Images for 8, 12, 16 Clients. 35

Table 4.10.Time for Saving the Hash Values in Database. 36

Table 4.11.Execution Time Versus. Number of Images for 4, 8, 12, 16 Clients, Using Multiple Copies of the Program. 40

Table 4.12.Dynamic Way Versus. Saving the Hash Values in Database. 41

Table 5.1.Bit Wise Comparison for Similarity Using 25,100,200,500 Images. 48

Table 5.2.Exhaustive Template Matching Using 25,100,200,500 Images. 49

Table 5.3.Comparing Between Exhaustive Template Matching and Bit Wise Comparison. 50

LIST OF FIGURES

Figure 3.1: Creating the Images Database Flow Chart. 18

Figure 3.2: Extracting the Hash Value Flow Chart. 19

Figure 3.3: Comparing the Hash Value Flow Chart. 20

Figure 3.4:Creating the Database. 21

Figure 3.5: Extracting Hash Value. 22

Figure 3.6: Specification of number of clients. 23

Figure 3.7: Client Form. 24

Figure 4.1: Time versus Number of Images for Sequential Comparison / Deletion for Base Image. 26

Figure 4.2: Time Verses Number of Images Sequential Comparison /Deletion for Random Image. 27

Figure 4.3: Speed-Up versus Number of Images (Second Experiment). 29

Figure 4.4: Efficiency versus Number of Images for the Second Experiment. 29

Figure 4.5: Speed- Up versus Number of Images for the Second Experiment. 30

Figure 4.6: Efficiency versus Number of Images for the Second Experiment. 30

Figure 4.7: Execution Time for Different Hash Algorithms. 32

Figure 4.8: Hash Algorithms versus Bit Wise Technique. (Execution Time). 34

Figure 4.9: Execution Time Versus Number of Images for 4, 8 clients. 35

Figure 4.10: Execution Time Versus. Number of Images for 8, 12, 16 Clients. 35

Figure 4.11: Processing of Images. 37

Figure 4.12: Execution Time Versus. Number of Images for 4, 8, 12, 16 Clients, Using Copies of the Program. 40

Figure 4.13: Execution Time Versus. Number of Working Copies for 2500 and 3000 Images. 41

Figure 4.14: Dynamic Way Versus. Saving the Hash Values in Database. 42

Figure 5.1: Query by Example Form. 44

Figure 5.2: Query by Example Module Flow Chart. 45

Figure 5.3: Query by Example with Options Form. 46

Figure 5.4: Bit Wise Comparison for Similarity Using 25,100,200,500 Images. 48

Figure 5.5: Exhaustive Template Matching Using 25,100,200,500 Images. 50

Figure 5.6: Comparing Exhaustive Template Matching and Bit Wise Comparison for finding similarity. 51

Chapter 1

INTRODUCTION

The number of images stored and applications developed for accessing images on the Internet has grown considerably in the last ten years. This causes many problems related with information retrieval on the Internet. Among a large number of images, it is often hard to find required images. There are three main problems that can be mentioned:

1) The naming problem
2) The description problem
3) The redundancy problem

Firstly, search engines are still using mainly metadata or keywords to create image databases. Metadata cannot deal with different meanings of words, and sometimes there may be no relation between the contents of the images and their names. For example, when one uses a camera for taking images, the camera generates names for those images automatically, with no relation with the image content. We call this the" naming problem".

Secondly, when the user doesn't know how to describe the image he/she requires, it is hard to find out the image he/she is trying to get. This will be referred to as the "description problem"

Finally, information redundancy consumes extra time in checking the results. Hence, there is a need for improving matching image display efficiency, this is called

the" redundancy problem". One way of improving display efficiency is the elimination of repetitions, which is the topic of this study.

Many studies have been performed for solving the three main problems discussed above. New search mechanisms and algorithms have been developed for more efficient image retrieval.

Content image retrieval mechanism is one such method. Content image retrieval appears as a way of solving the naming problem stated above. The content- based retrieval method works by considering the low level features of multimedia files.

Ontology based retrieval method is one technique for content image retrieval. The Ontology based method uses Meta data and some keywords, Hybrid methods can also be used, combining the two methods mentioned above.

On the other hand, new ranking algorithms were developed to find matching results in a short time. Those algorithms take into account the multimedia content of the website in the ranking process .The aim of the new ranking algorithms is improving the chance of finding multimedia files through the internet.

Query by example method was developed to solve the description problem mentioned above. This method is efficient when the users have some images and they want to get similar images. The user uploads the image at hand and the search engine tries to find similar images. Lately, query by sketch method was developed to increase the efficiency of the query by example technique. Query by sketch works using the same techniques as query by example, but with more options. For example, query by sketch allows the user to employ drawing tools to describe the expected image.

Lately, various software packages were developed using these new mechanisms, to improve the performance of search engines. One example is the flexible and

extensible framework for web image retrieval mechanism (FGWIM) [8]. FGWIM works using high level semantics and low level visual features of images for extracting information from files.

Document clustering can also be used to solve the naming problem when data is clustered, and similar web documents can be found more easily using search engines.

Considering the redundancy problem, up to our knowledge, there is no research on eliminating the repetition of the same result in search engine outcomes. The main objective of the work presented in this thesis is to improve the efficiency of search engines when dealing with images, by eliminating repeating images.

We propose a new method for the elimination of repeated occurrences in image search engines. We have developed software that can create an image database. Then, it calculates hash values for the images. Finally, it compares the hash values to find repetitions, and marks only one copy of repeating files for further use.

To make the proposed method more efficient, we allow copies of our software to process information in parallel. In this case, the number of images in the database is divided evenly between the parallel copies. The system administrator decides on how many copies should be run depending on the total number of images in the database.

Then, we have developed another module, which works similar to a query by example search engine. This module can be used for cases where the user has an image, and is looking for its copies, or images similar to it.

The other parts of this thesis are organized as follows: Chapter 2 discusses related studies. Chapter 3. Explains our technique for eliminating of repeated occurrences of image. Chapter 4 outlines performance studies. Chapter 5, discusses finding similar images by employing the query by example technique. Chapter 6, states some conclusions and containing discusses further work.

Chapter 2

RELATED WORKS

2.1 Overview of Internet Search Engines

Search engines collect descriptive information from websites. This information mainly contains keywords. Most search engines use the spider technique to collect this information. After the descriptive information is collected, the next issue is to analyze this information using special algorithms like finding the percentage of the number of hits of the website. After that, a database, which contains the keywords, the website address, images and information about the website, is created. One main problem is that, on the Internet, many websites have copies of the same images, which means an unnecessary effort will be employed when searching.

When the user executes a query as for the retrieval of a certain image, search engines will run it, and show the user a list of images. This list will sometimes contain thousands, or even millions of images. In such a case, it takes a lot of time for the user to check all the images listed by the search engine. The following table, which is the result of our own study, gives an idea on the total number of images reachable by the Google search engine.

Table 2.1.Number of Images For Some Queries (Reachable By Google).

Search Keyword	The number of images(reachable by Google)
images	189,526,563
*.jpg	2,147,483,647
*. jpeg	19,991,129
*.gif	584,742,791
*.png	468,217,403
*.ico	9,005,572
Total number of images	3,418,967,105

The website which has the highest rank will show at the beginning of the list of results. The ranking of a website depends on the number of hits, the keywords, website Meta Tags and the content of this website [11]. In order to keep the ranking position of websites, we do not physically delete repeating images. Instead, a flag field is added to the database. For the first one of repeating images we set it to one, for all others we set it to zero.

2.2 Overview of Related Work

Multimedia searching has become an important research field these days. Many researchers are trying to improve the efficiency of getting Multimedia files through the Internet. Initially, researchers studied the current search engine mechanisms. Accordingly, new search mechanisms and algorithms were developed for similarity. In this chapter, we shall first study the mechanisms of popular search engines.

2.2.1 Studies on Current Search Engine Mechanisms for Finding Images

The challenges that make using Web-scale multimedia search engines difficult were studied by other groups of researchers [4]. The first challenge is extracting useful features to represent multimedia files .The second challenge is the difficulty of finding an appropriate similarity measure. For example, “images taken in different

lighting condition can display different features after extracting its features" [4]. The third challenge that restricts the deployment of large scale systems is that multimedia search engines must be able to scale well with respect to both data dimensionality and data quantity. In addition, identifying key features in images is easy when a human detects the key features, but it is hard when it is done automatically.

A study of the functionality of multimedia search engines was conducted by examining 102 web search engines in [6]. There were several issues to check: (1)Find the number of Web search engines that support multimedia searching, (2) find the functionality and methods offered in multimedia search, such as ''query by example'', and (3) the support for personalization or customization as advanced search options.

The study indicates that there are 65 general purpose engines and 37 multimedia search engines. 43 out of 65 general purpose search engines support text media search only. All web search engines still rely on file meta data, such as file format, size and characteristic of the web site content. Image retrieval by contents is very limited; only 5 out of 102 web search engines support this mechanism. Even when content-based retrieval is supported, low level features are used. Low level features extract file properties like texture, size, or colours. Web search provides limited multimedia search functionality, query by example is still not available for the users. Support for personalization or customization is too limited.

One group of researchers studied searching for digital images on the web [1]. They investigated how users structure image queries, and the current image retrieval approaches. The current image retrieval approaches are concept-based content and content based retrieval or a concept-content combination. They compared the existing image query classification schemes. None of the three classification schemes

captured the richness of web image searching. Also, they found that the main problem was the generation of file names randomly or by using temporal character sequences, during the creation of image databases that makes using the current image retrieval approaches not suitable for multimedia. Moreover, they found that multimedia search engines use same mechanisms as textual information search engines. Metadata is often insufficient when dealing with multimedia content. Digital images are increasing the need for more effective methods of searching, and retrieving image data. They suggest comparisons and additional classifiers for web image searching as a way to improve the efficiency of search engines [1].

In [16], there is a study conduct to check the current search engines and their mechanisms, finding they are good to retrieve images or not. They divide the current search engines into three types:

1) Search engines with a large image database.
2) Experimental search engines.
3) Meta-search engines.

Google and Yahoo are examples of first type image search engines, which have a large image database. These databases are created by indexing the keywords and the images.

Second type at image search engines is specific image search engines for indexing images or multimedia like Corbis & Getty Images. These websites are often experimental and have limited databases that are restricted by size when compared with sites such as Google.

Finally, there are image search engines that are called Meta-Search engines, which send users' requests to multiple search engines and then display the 'multiple' results. Image Search Mechanisms were also studied in [16].

Most of search engines ask the user to type a keyword and then compare it with the content of their database, using the file type that helps to detect the desired type of files, e.g. jpg or bmp format. Then the search engine displays the result. This method is good for large databases, but it is not suitable for multimedia files, for example, in Google or Yahoo.

The Second Mechanism is the creation of the database by a human. The database builder will build categories and put the images on it (e.g. cars group, flowers...). However, as we know there are millions of images on the internet. Therefore, it is too difficult to determine major categories and to build this type of a database. It is more difficult to keep it updated.

The research group have performed three experiments to compare the performance of some search engines: The first experiment uses one word size test queries. The second experiment uses two word size test queries. The third one uses three word size test queries. The experiments were performed on image search engines such as Google, Yahoo, Ditto, Corbis, Web Seek, Getty Images Creative, Picsearch, and Ithaki. The results are as follows: The average precision is 55% for the first experiment, 50.6% for the second experiment, and 20.7% for the last experiment.

As a conclusion of their work, they report that, most search engines are indexing images using text and they rely on keyword based images searching. [16].

The effect of the number of query words on image search engines was studied in [17]. “Word Tracker periodically compiles a database of over 330 million search terms which is updated on a weekly basis. All search terms are collected from the major meta crawlers such as Dogpile and Metacrawler “[17].This group have performed experiments on four search engines (Google, Yahoo, Msn, and Ask). They

selected fourty queries from the list of Word Tracker [23], and categorized them into four groups of queries: one word, two words, three words, and four words. Then, first twenty results of each query were judged if they are relevant or not by two humans. They have done the performance evaluation of image search engines in terms of precision and normalized recall. Precision is defined as the percentage of relevant documents to the search out of all retrieved documents. Recall is the percent of relevant documents which are successfully retrieved [19].

They found that Google has the lowest number of relevant image items. The performance of Google is also the lowest for one-word queries. On the other hand, the average ratios of performance for Ask, Yahoo, and Msn are lower than that of Google's for two-word, three-word, and four-word queries. Google retrieved more relevant items than other search engines when the number of query words increases. In short, Google appears to be the best image search engine. In general the search engines give a good result for one word queries, and performance is decreased when the number of words in queries increasing. [17].

2.2.2 Studies for Improving the Efficiency of Search Engines

Lately, new software was developed by researchers to improve the performance of search engines in finding multimedia files on the Internet. Some of those studies will be mentioned here.

2.2.2.1 Flexible and Extensible Framework for Web Image Retrieval

A flexible and extensible framework for web image retrieval (FGWIM) was presented in [8]. FGWIM works using high level semantics and low level visual features of images through extracting information from files. It extracts information in several levels and images are considered as a part of the web site. So, the images

should not be specified only by images themselves, but also with respect to the web contents surrounding the images. In FGWIM, special techniques and components like relevant feedback mechanism and data mining for knowledge discovery is used. As a result, search engine performance for multimedia content retrieval is improved [8].

2.2.2.2 Direct Searching of Video Content (DIVAS)

A method for direct searching of video content without using metadata information was presented in [11]. DIVAS work is based on the finger printing method and MPEG. For video characterization, features of several classes are used. In the first class there are features that make some sort of segmentation. Segmentation means logical division of long video sequences into several smaller sub sequences. At the first stage, extract key frames are used. Then, average of the colours of each I frame are extracted. Then these properties are saved in database as finger print for that video. After the user uploads the video file, DIVAS will extract its properties and will try to find the same files in the database. This method can help people for finding videos when they have a clip of that video. DIVAS can be considered as a query by example search engine. [11].

2.2.2.3 SCENIQUE

SCENIQUE is a program for managing images by using visual features (e.g., colour and texture) [9]. SCENIQUE was proposed as a multifaceted image search and browsing system. It uses both visual features and tags. SCENIQUE has tools to manage the image collection. It, stories the feature vectors that are automatically extracted from photos, and a Tag database which keeps the current tags in each tag tree.

The Interface of SCENIQUE is as follows:

1. Facets construction: Facets construction is supported by an intuitive interface that requires the user to set the name of the dimension.
2. Photo annotation: For annotating an image, the user selects a photo together with a dimension of interest.
3. Search facilities: used to search the photo collection.
4. 3-D browsing: Photo collections can be explored by the user through an intuitive browsing interface.

Using this tool gives one an opportunity to manage images more efficiently. [9].

2.2.2.4 Lazy

In [2]. Lazy program is discussed. Lazy uses a Content-Based Image Retrieval (CBIR) system that combines dynamic, user-driven search capabilities. Lazy system improves query-by-sketch and query-by-example by using intelligent User Interface Agents (UIAs). The UIAs use both neural networks and an expert reasoning system to help with relevant feedback. In addition, a new CBIR evaluation metric was presented. Lazy has four different types of user interfaces in CBIR systems to resolve image queries: keyword searching, category browsing query-by-example and query-by-sketch. Also, there is a thumbnail browsing, option which works on creating groups that contain all files related with it. For example, one can create a group which contains all files related to cars. Then inside the cars group, you can create sup groups with more detail like one group for each car brand [2].

2.2.2.5 Query by Example

A view- based web page retrieval system developed in [2, 3], enables a user to search web pages using a “visual query”. This method, called “query by example”, is

more powerful when one wants to get files similar to what s/he already has. In this technique, when a sample file is uploaded, search engines try to find similar files [2, 3].

2.2.2.6 Query by Sketch

Another method called “query by sketch” is developed to improve the performance of the query by example method [2, 3]. Query by sketch searches web pages using a visual query, and it mainly gives the user more options like using drawing tools for describing exactly what is required. The system uses “query by sketch” to give some information about what the user wants. Then it will evaluate the similarity between web pages and the sketch, using an EMD-based method.

EMD is a matching algorithm to compute distances between the colour histograms of two digital images. Sketch works also through drawing tools, and can ask the user to draw what s/he wants [2, 3].

2.2.2.7 Hybrid Methods

One of the new mechanisms proposed uses a Hybrid method, which was presented for effective searching through multimedia content (2D/3D image and video) [7]. The search engine developed in this method uses three ways for executing the queries: The ontology-based method, the content-based method, and the hybrid method.

The ontology-based method uses the Meta data and the keywords. The content-based method works by extracting the low level features from the multimedia files. The hybrid method uses the other two methods during the query execution. One project developed by this group is REACH, which works on hybrid retrieval. It was

tested on a museum database. Results show that a hybrid approach improves the chance of getting the correct file by a query [7].

2.2.2.8 Automatic Ranking of Websites

Ranking websites is basically ordering the websites in the list displayed as the result of a search query [14]. Ranking websites affects the order of results. The ranking of a website depends on the number of hits, the keywords, website Meta Tags and the content of this website [11]. The website with a high rank will show at the beginning of the list of results. However, this may be unfair with multimedia files. The images on the Web are an important part of web contents. Both text and image content can contain useful information that should be used in retrieving web images. A group of researchers implemented an automatic ranking process, working on integrating the keyword and visual features for web image retrieval. The web image retrieval system named VAST (VisuAl &SemanTic image search) was prepared as a result of their studies. In general, after users execute a query, the algorithm works on the result of the query by checking it and ranking it depending on the multimedia content. Then it displays the results for the user [14].

2.2.2.9 Key Block

"Key block" is a new approach termed for content- based image retrieval [15]. Key block is a generalization of the text-based information retrieval technology in the image domain. In this approach, methods for extracting comprehensive geographic image features are provided, which are based on the frequency and correlation of representative blocks that are termed "key blocks" of the geographic image database. Features are extracted information from the images in three stages. The first stage generates code books that contain key blocks of different resolutions

by dividing images into smaller blocks. Then subsets are selected. Secondly, images are encoded. Each image in the database will be decomposed into blocks, then for each one of these blocks the closest entry in the code book will be found and an index will be stored (each image is considered as a matrix). The third stage is image representation and retrieval, it extracts comprehensive image features, based on frequency of the key blocks within the image [15].

2.2.2.10 Document Clustering

Document clustering is a technique can be used to find similar web documents out of the documents obtained by search engines. Web documents can be organized by using clusters, which leads to a categorization of the data. Then we can find the relevant web documents quickly. Clustering techniques can be divided into hierarchical and partitional methods [18].

Hierarchical methods produce a sequence of nested partitions, Hierarchical methods can be divided to two methods, agglomerative and divisive. Agglomerative methods start with one-document clusters, and recursively combine the most suitable clusters. Divisive methods start with one cluster that contains all the documents, and recursively divides it into suitable clusters. Some Clustering algorithms that belong to hierarchical methods, are HAC (Voorhees, 1986), STC (Zamir & Etzioni, 1998), and DIVCLUS-T (Chavent, Lechevallier, & Briant, 2007) [18].

Partitional methods work by dividing the entire document collection to a specific number of clusters. The main aim of Partitional methods is to achieve high intra-cluster similarity. Some clustering algorithms that belong to partitional methods, are K-means (MacQueen, 1967), SRE (Zha, He, Ding, Simon, & Gu, 2001), and k-Attractors (Kanellopoulos, Antonellis, Tjortjis, & Makris, 2007) [18].

One clustering algorithm was presented in [18], called On-The-Fly Document Clustering (OTFDC). It generates a set of clusters from other web search results. This method finds similar clusters using different ways. One approach is checking if the clusters have a semantic relation. Semantic relations can be one of the following three:

a) Equivalence: the clusters are equivalent if they are at the same level. For example, (“home”/ “house”).
b) Hierarchy: the first cluster can be considered as a group or set, and the second cluster as a subset or part of the group. For example, (“fruit”/ “apple”) and (“vehicle” / “car”).
c) Association: in order to be associated, clusters should not be equivalent or hierarchical. “The clusters are semantically associated to such an extent that the relation between them should be made explicit. For example, (“flour” / “wheat”)” [18].

The advantages of On-The-Fly Document Clustering:

(1) It can be applied to multilingual web documents.

(2) It improves the clustering performance of any search engine. (They simulated the combined search engines:”Google-OTFDC”, “Yahoo-OTFDC”, and “Vivisimo-OTFDC”).

(3) OTFDC does not need any predefined information on the distribution. (Unsupervised learning)

(4) Clustering results are generated on the fly, and fitted into search engines. This means OTFDC is a recursive algorithm, and it still generates candidate clusters on the fly, in response to a user query. [18]

Chapter 3

ELIMINATION OF REPEATED OCCURRENCES IN IMAGE SEARCHING

In this chapter, software design issues will be discussed, including the programming environment, the database issues, basic algorithms, and the user interface.

3.1 Programming Environment

In this section we are going to discuss the programming environment, in which, the software for this thesis is developed .We have built the software using “VB.NET (2008)”. VB.NET has many advantages, like support for graphic user interface, and support for hash algorithms. VB.NET also has the ability to create client-server applications.

As for the hardware, we used a server PC which has a core 2 duo CPU of 1.83 GHz clock frequency and 3.00 GB of RAM. We have installed the Windows 7 OS environment on the server.

3.2 The Database

We have created a images database as a part of our work. We used “SQL Server (2008)” for creating the database. We saved images with their information in the database.

We selected SQL Server for creating the database, as it supports VB.NET. Secondly, SQL Server offers good security control for our database. Finally, saving a huge number of images inside the database is possible.

3.3 Software Mechanism

The software developed for comparison / deletion of images can be described in three stages as follows:

3.3.1 Creating the Images Database

In creating the images database, our program extracts the properties of images. Then, it saves the images with their properties in the database.

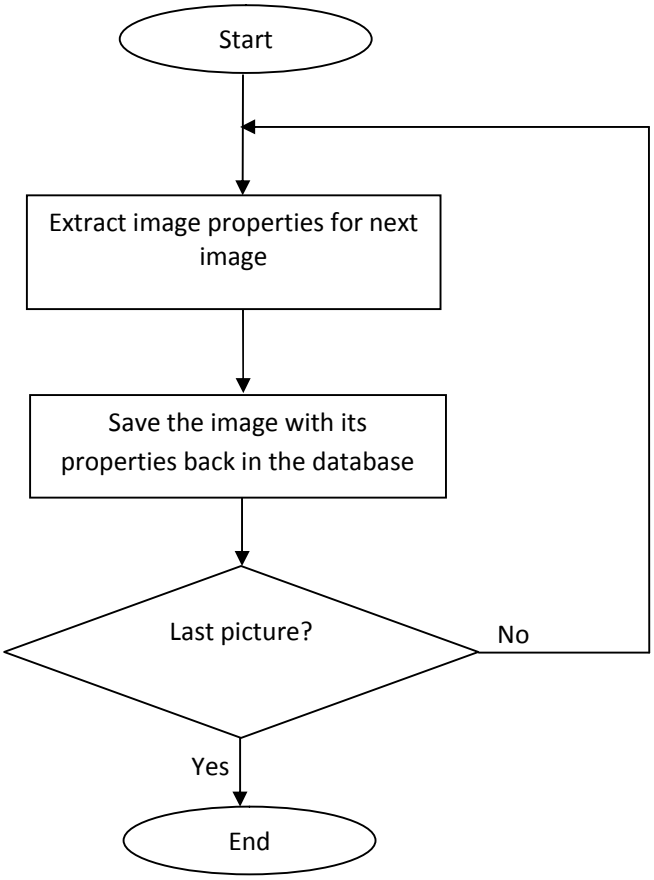

Figure 3.1: Creating the Images Database Flow Chart.

3.3.2 Computing the Hash Value

Firstly , the hash value comparison program will convert an image to an array of bits. This array will be the input for the MD5 hashing algorithm which is discussed detail in chapter 4. Sixteen unique bits will be the output of MD5 for each image. Then the software will save this hash value in the database togather with the image.

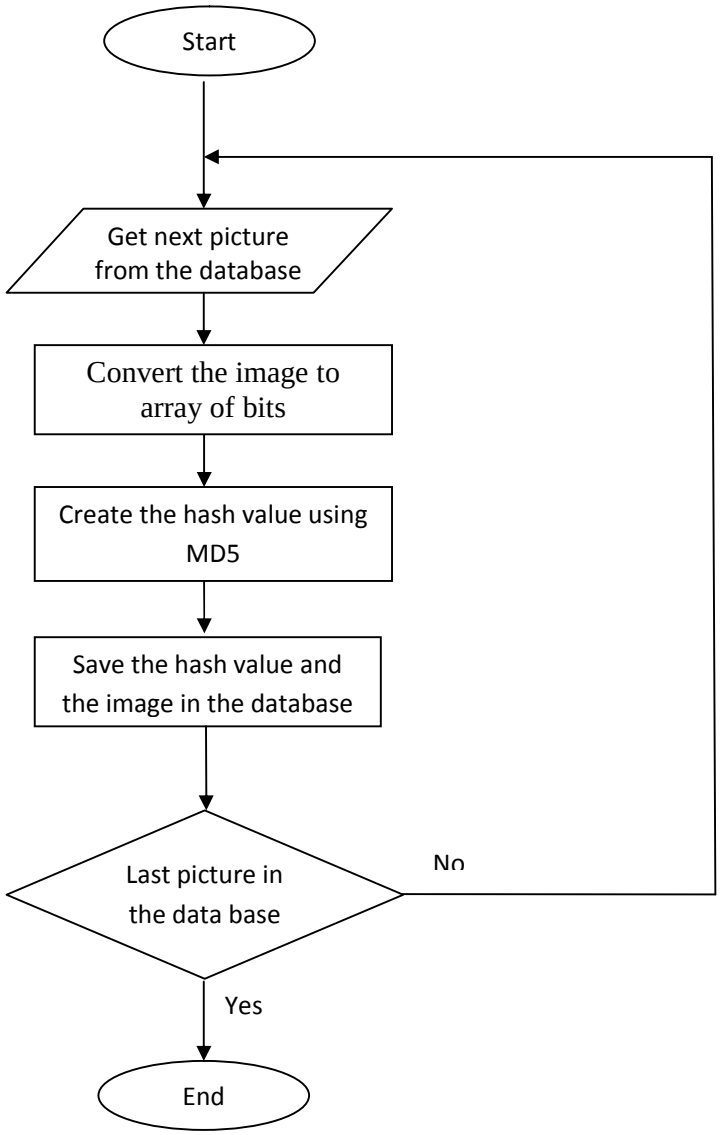

Figure 3.2: Extracting the Hash Value Flow Chart.

3.3.3 Comparing the Hash Value

The comparison program will get the hash value for the selected image from the data base .and compare it with the hash values for repeating images. If repeating images are founded, the program will keep the first image's flag as one and set flags for the repeating (i.e. second, third, etc.) images to zero.

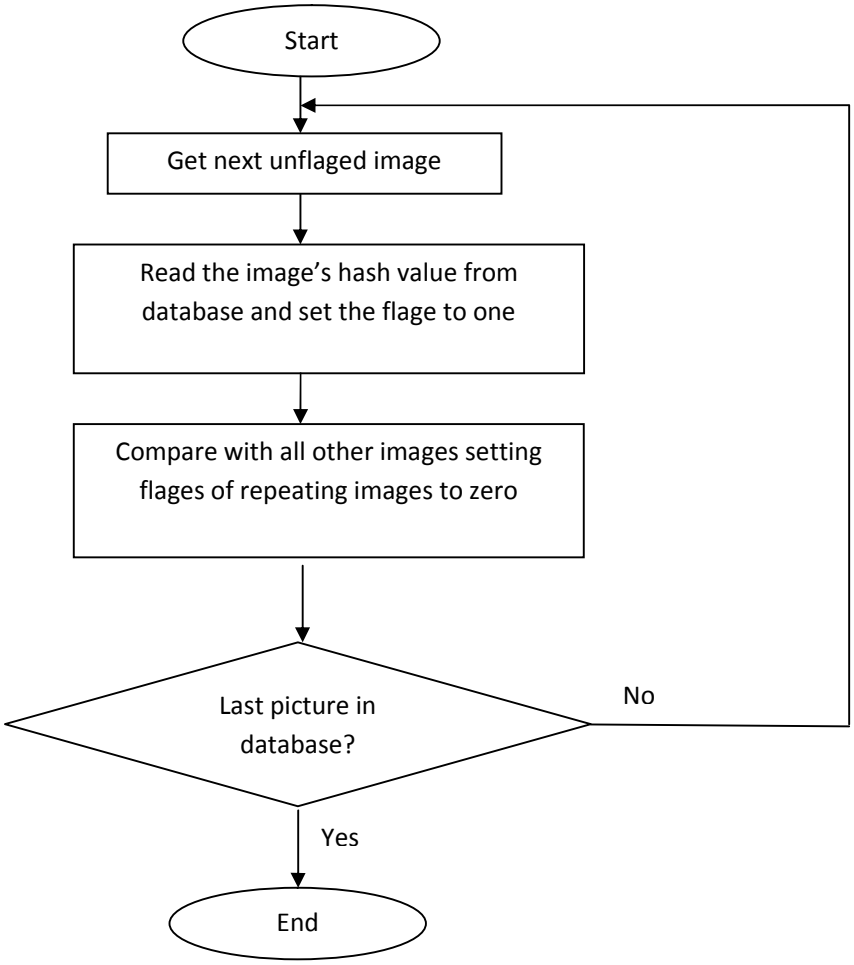

Figure 3.3: Comparing the Hash Value Flow Chart.

3.4 User Interface

The Software developed in this study has an administrator interface and a (client) user interface. The Administrator Interface allows the system administrator to create the database. Figure (3.4). Shows the administrator interface form for creating the database.

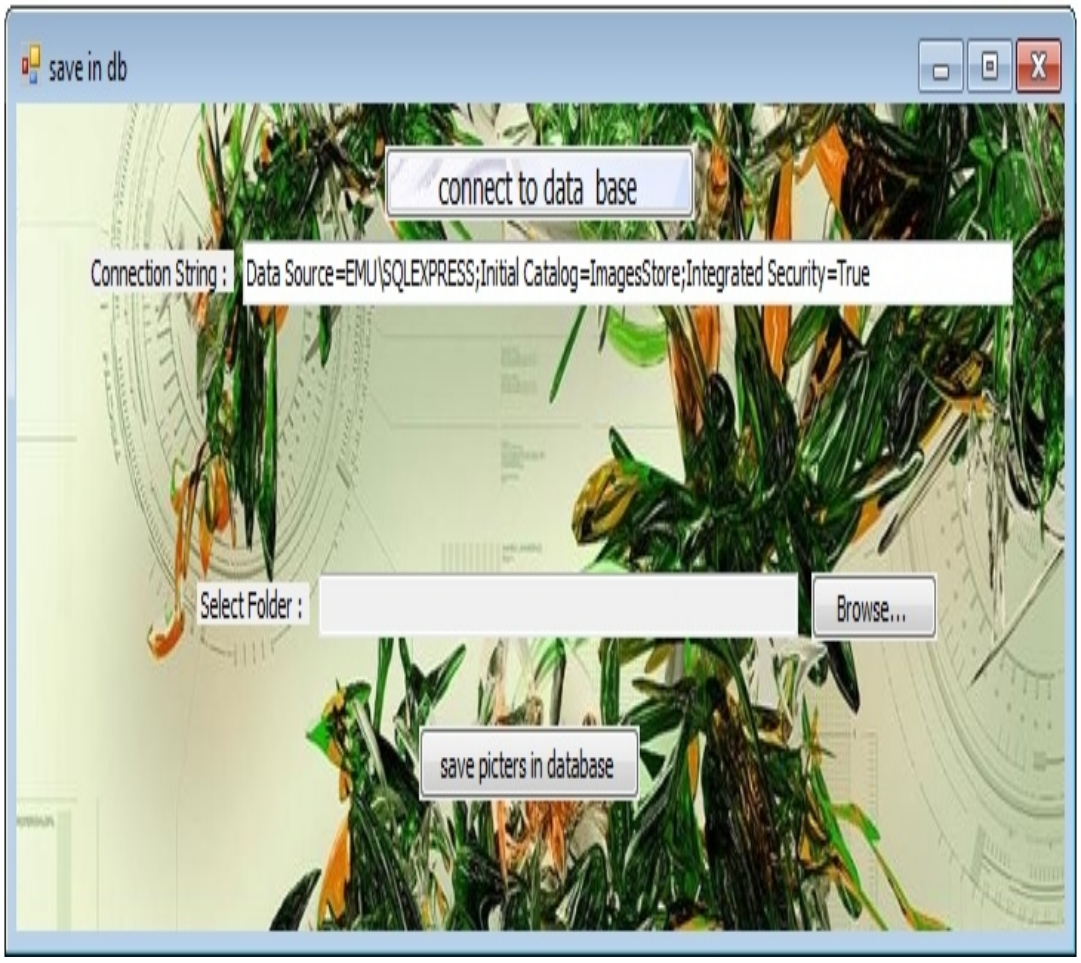

Figure 3.4:Creating the Database.

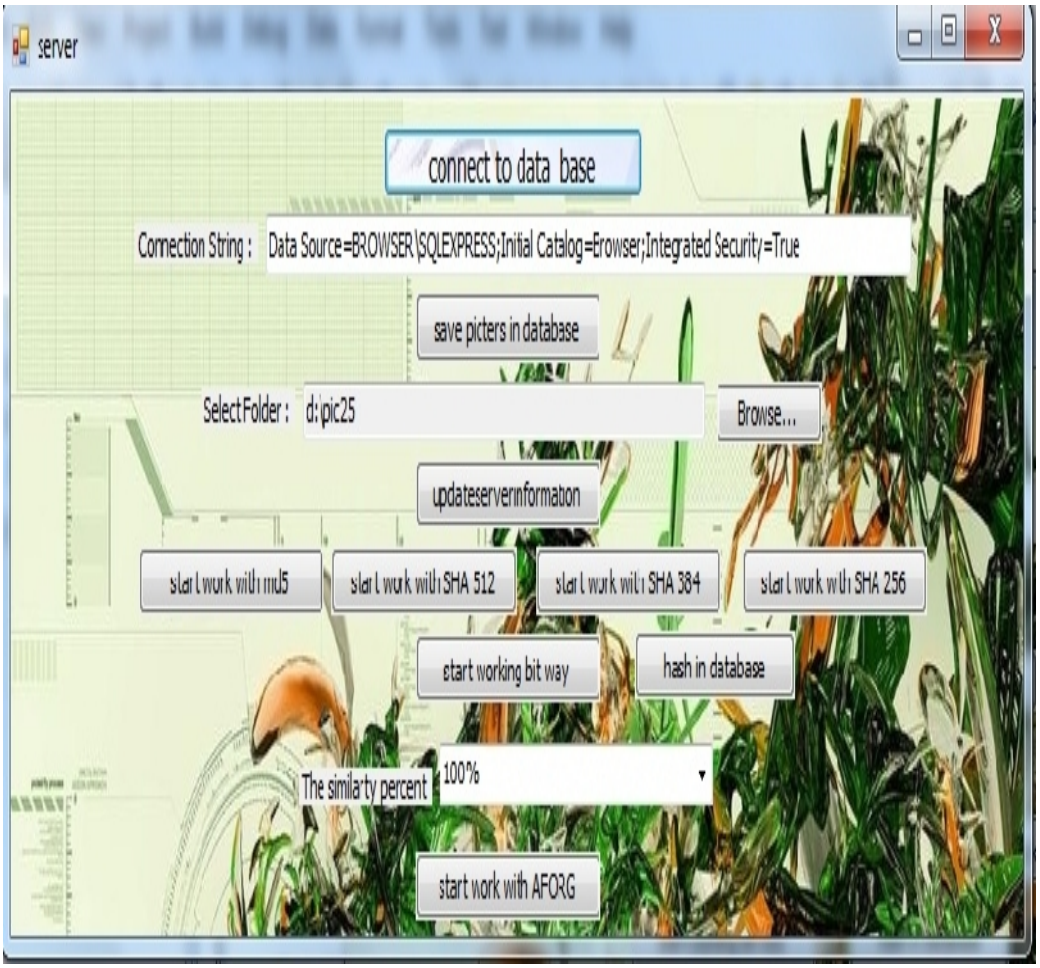

Figure 3.5: Extracting Hash Value.

The second form of Administrator Interface allows the system administrator to compare the images using all hash algorithms mentioned in our thesis (MD5, SHA512, SHA256), or bit ways comparison can be used. Figure 3.7 shows the administrator second form.

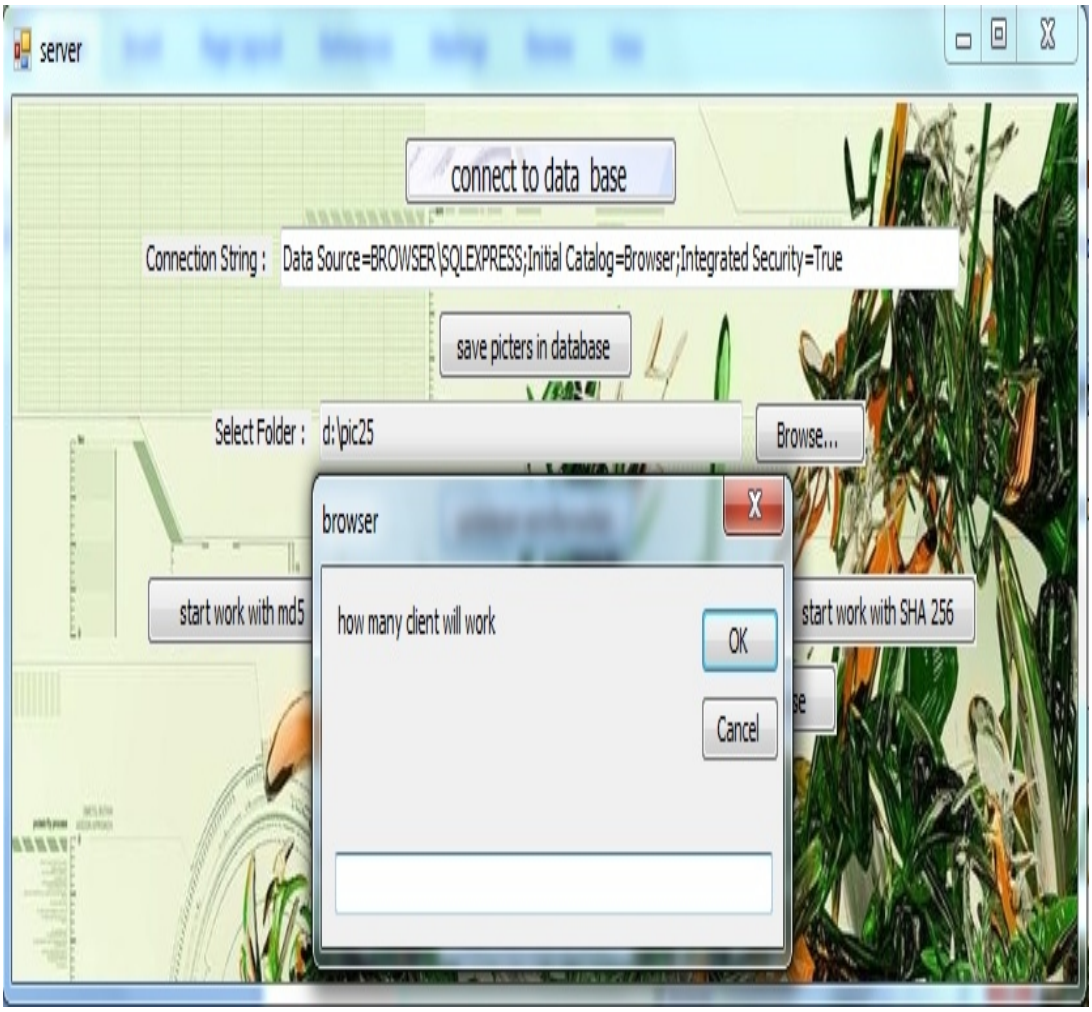

Figure 3.6: Specification of number of clients.

The system administrator specify how copies of the client working. If copies are to be started, the software will divide the number of images between the working copies. Figure.20 shows the related interface.

The (Client) User Interface

The client uses this form for saving the client information, to read information from the database and to start comparing the images.

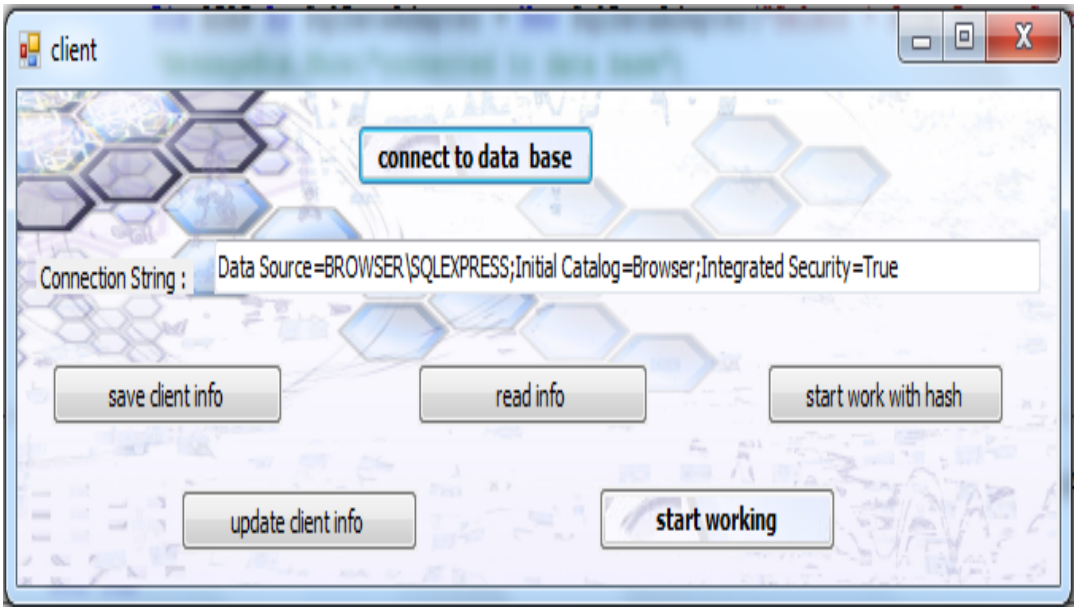

Figure 3.7: Client Form.

Chapter 4

PERFORMANCE STUDIES

4.1 Introduction

We have conducted some experiments to test the performance of image comparison using different techniques. This chapter outlines the details and the results of performance studies.

4.2 Bit-Wise Comparisons

At the beginning, we have selected the" bit- wise" comparison technique to compare images. Bit- wise comparison compares all the pixels of two images one by one. If all pixels in both images are the same, only one of those images will be considered in later searches.

To see the effect of using bit-wise comparison, we have performed some experiments.The first experiment was conducted on an artificial database, created in two different ways:

In the first approach, the images in the database are created by taking copies of seven "base images". Each one of those base images is then copied many times in order to get a specific total number of images in the database.

In second approach, the images are created by using randomly chosen images from internet search engines. Then, the same copying process as in the first group was applied.

4.2.1 Sequential Execution

Sequential execution means only one copy of the program works at a given time. The software will take one image and compare it with all images in the database sequentially .In case the next image from the database is the same as the "comparator", it deletes this image. Table 4.1 and Table 4.2 give the results of the bit wise comparison technique for two different database construction approaches.

Table 4.1.Results of Sequential Comparison / Deletion for Base Image.

Number of images in the original data base	# of deleted images after executing the algorithm	Remaining images in the database	Time sequential work(seconds)
25	18	7	19
50	43	7	40
100	93	7	83
500	493	7	475

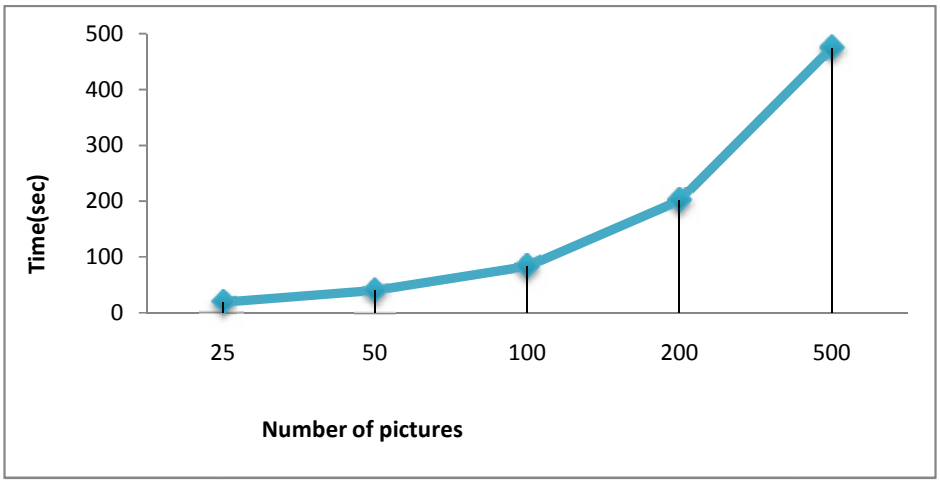

Figure 4.1: Time versus Number of Images for Sequential Comparison / Deletion for Base Image.

Table 4.2.Results of Sequential Comparison / Deletion for Random Image.

Number of images in the original data base	# of deleted images after e the algorithm	Remaining images in the database	Time sequential work (seconds)
25	9	16	22
50	27	23	82
100	71	29	164
500	291	209	850

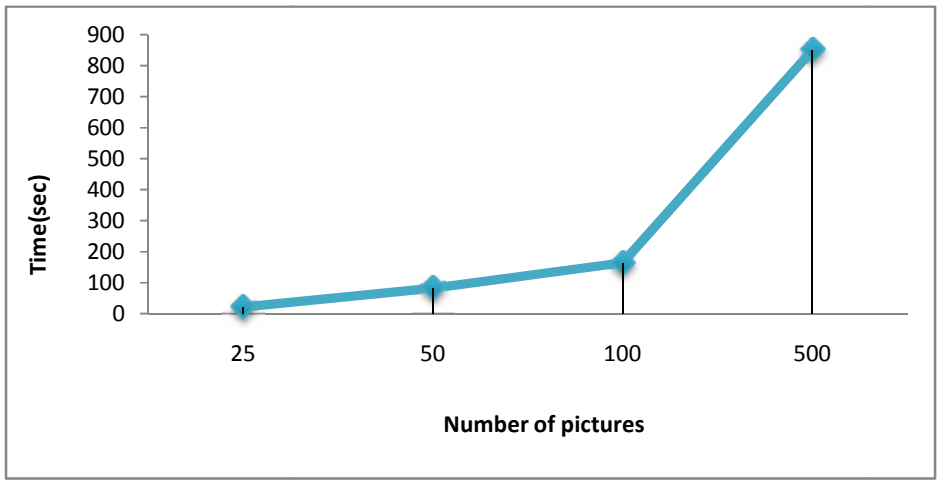

Figure 4.2: Time Verses Number of Images Sequential Comparison /Deletion for Random Image.

From these results, it is clear that bit wise comparison needs a long time to compare even 500 images. In real life, an image database will contain millions of images, so the efficiency of bit-wise comparison technique will be very low.

4.2.2 Parallel execution: Client - Server Architecture

After the first experiment, we have started to think about a more efficient way to do these comparisons. One idea might be using a parallel mechanism. We prepared a software module that uses the client- server architecture. This client- server system works on the same database in parallel.

We performed the second experiment to see the efficiency of this client – server method. The results of our second experiment are given in Table 3 and Table 4. The first group of images in our second experiment is the same group of images as the first experiment. The second group of images is the same as the second group of images in our first experiment.

After preparing the database, we divided it into two parts. One part is checked by the server, and the other is checked by the client. The results show the improvement of using a parallel search, which means the server and the client will work together. Speed- up is obtained by dividing the execution time for the sequential case, by the execution for the client-server method. Efficiency is obtained by dividing the speed up by the number of working processors.

Table 4.3.Results of the Client – Server Method for Base Images.

The number of images in the original data base	**number of deleted images**	**Remaining images in database**	**Time**		**Speedup** $S_p = \frac{T_1}{T_p}$	**Efficiency** $E_p = \frac{S_p}{p}$
			parallel Work (second) T_p	**Sequential work (second)** T_1		
25	18	7	10	19	1.9	0.90
50	43	7	23	40	1.73	0.865
100	93	7	50	83	1.66	0.83
500	493	7	300	475	1.58	0.79
1000	993	7	760	1046	1.55	0.795

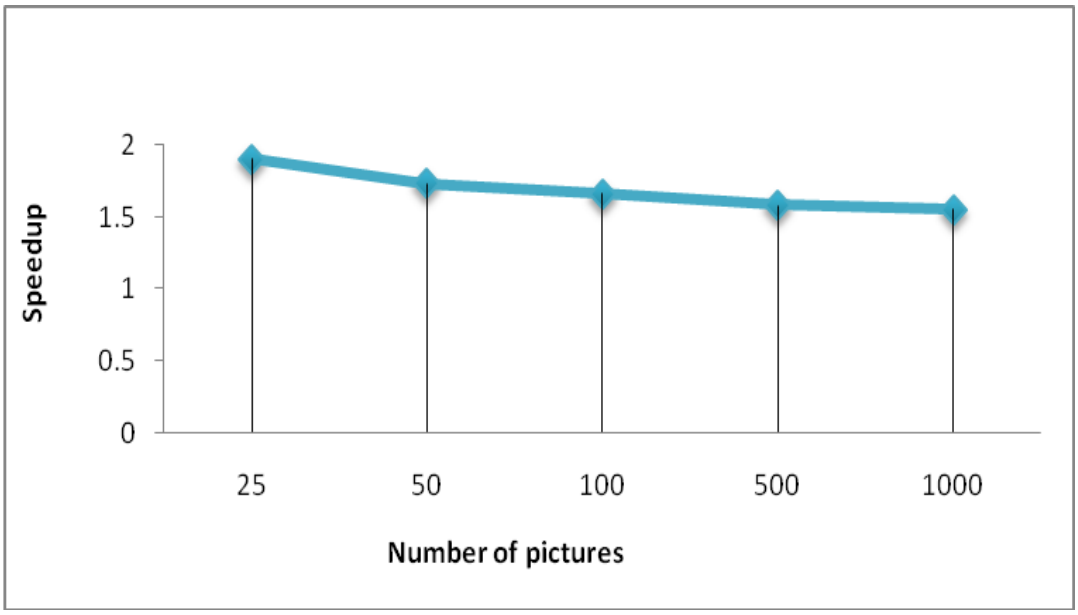

Figure 4.3: Speed-Up versus Number of Images (Second Experiment).

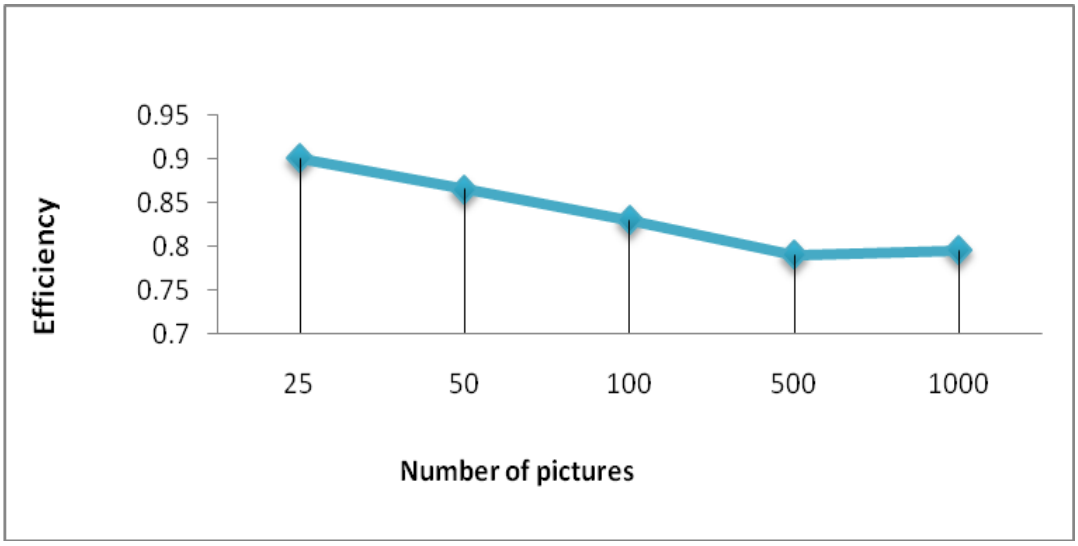

Figure 4.4: Efficiency versus Number of Images for the Second Experiment.

Table 4.4.Result of the Client –Server Method for Random Images.

The number of images in the original data base	number of deleted images	Remaining images in database	Time		Speedup $S_p = \frac{T_1}{T_p}$	Efficiency $E_p = \frac{S_p}{p}$
			parallel Work (second) T_p	Sequential work (second) T_1		
25	9	16	15	22	1.5	0.733
50	27	23	55	82	1.49	0.735
100	71	29	113	164	1.46	0.730
500	291	209	579	850	1.4	0.734

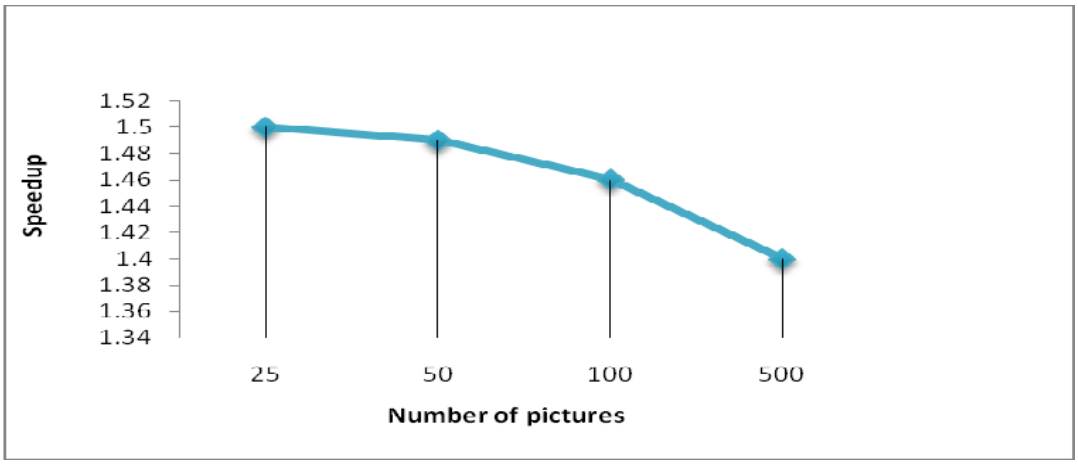

Figure 4.5: Speed- Up versus Number of Images for the Second Experiment.

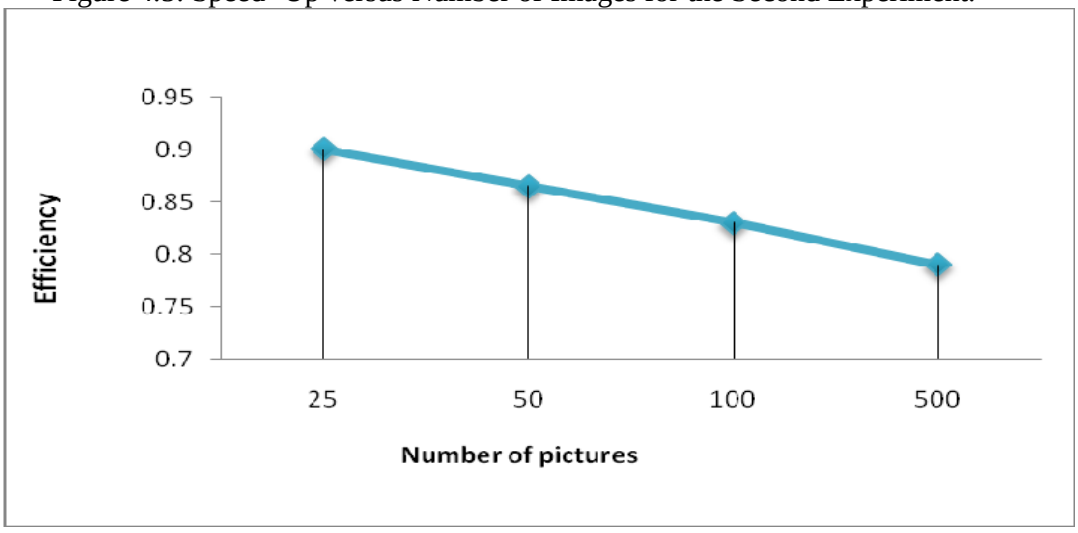

Figure 4.6: Efficiency versus Number of Images for the Second Experiment.

In Tables 4.3 and 4.4, we observe a slight improvement in our parallel method. Nevertheless, it still needs a long time to compare the images in the database.

Considering the inefficiency observed in both methods, we decided to use a hash technique for comparing images.

4.3 Hash Comparison

A hash algorithm is a cryptography function that takes any information as input and converts it to a numeric code. The outputs of these algorithms are unique for each file, and it is like a fingerprint. Using hash algorithms, we can compare files with less amount of data. Each image has a unique hash value, we can compare this hash value for images. [12, 13].

Hash algorithm types:

Various hash algorithms were considered for the study. Those are:

a) SHA: The Secure Hash Algorithm (SHA) was developed by NIST and is specified in the Secure Hash Standard (SHS, FIPS 180). SHA-1 is a revision to this version and was published in 1994. It is also described in the ANSI X9.30 (part 2) standard. SHA-1 produces a 160-bit (20 byte) message digest. [12].

b) MD5: MD5 was developed by Professor Ronald L. Rivest in 1994. Its 128 bit (16 byte) message digest makes it a faster implementation than SHA-1. [12].

The following table shows some properties for different versions of SHA and MD5:

Table 4.5.Comparison of SHA and MD5.

properties	SHA 256	SHA 384	SHA 512	MD5
Message size/bit	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$	∞
Block size/bit	512	1024	1024	512
Number of steps/bit	128	192	256	64

As stated before, the outputs of hash algorithms are unique for each file. It is like a fingerprint. This advantage gives us a chance to use hash algorithms for comparing the images to check if they are the same or not. We conducted a number of experiments to see the effect of various hashing technique. Table4.6 outlines a comparison of execution times for different hash algorithms.

Table 4.6.Execution Times for Different Hash Algorithms.

Number of images in the original data base	**Time(seconds)**			
	SHA 256	SHA 384	SHA 512	MD5
25	19	25	26	7
50	27	33	35	15
200	68	74	75	56
500	157	163	170	145

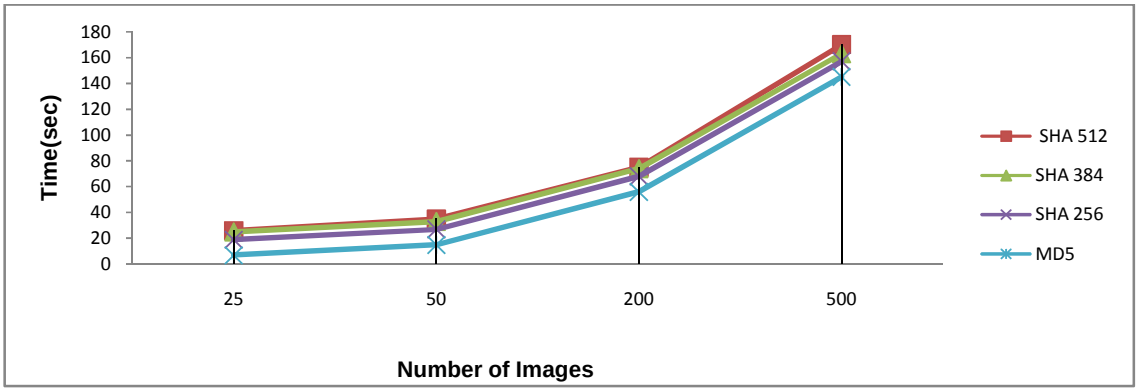

Figure 4.7: Execution Time for Different Hash Algorithms.

Looking at the search time results in Table 4.6, we decided to chose MD5, because of its advantages: the message size can be infinite and, the hash value is small in size (16 bytes) compared to other hash algorithms.

4.4 Comparison of Hash Algorithm and Bit-wise techniques

In this section, we outline a comparison between the bit- wise comparison and hash algorithms methods. Hash algorithms are more efficient than a bit wise comparison. Using hash algorithms, we need to compare a limited number of bits only, but in using bit- wise comparison, we compare the number of pixels in width multiplied by number of pixels in height. Using hash algorithms, we can find only the images which are 100% similar to each other, but using bit wise comparison, we can find images with any percentage of similarity.

For instance, we can use the bit wise comparison program to find the images which are similar to given image with a percentage of similarity 50% or more. Table 4.7. Comparison of the execution time results of hash algorithms and bit wise comparison.

Table 4.7.Execution Time of Hash Algorithms and Bit Wise Comparison Technique.

Number of image in the original data base	**Time**	
	hash algorithm (MD5) (seconds)	**bit wise comparison (seconds)**
25	7	19
50	15	40
200	56	83
500	145	475

Comparison of bit wise and hashing approaches shows that the hashing technique is much faster than the bit wise comparison technique, especially, for large numbers of images in the database.

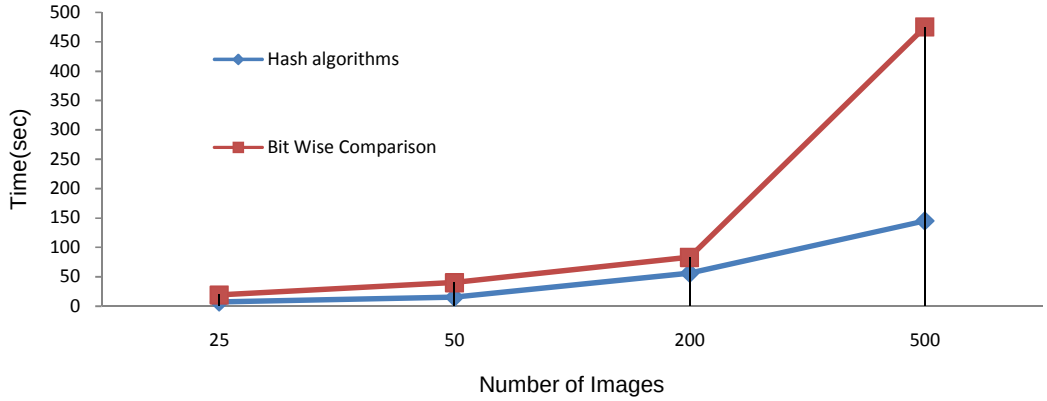

Figure 4.8: Hash Algorithms versus Bit Wise Technique. (Execution Time).

4.5 Parallel Work with Hash Algorithm

We performed another experiment in using the hash algorithm technique. In this experiment, we used more than one client. Therefore, we can divide the work on different clients, and as a result we will save time. The execution times for 4 and 8 clients are given in Table 4.8.

Table 4.8.Execution Time for 4 and 8 Clients.

Number of images in the original data base	Execution Time- Using four clients (seconds)	Execution Time- Using eight clients (seconds)
25	7	7
50	17	15
100	22	18
200	26	20
500	33	23

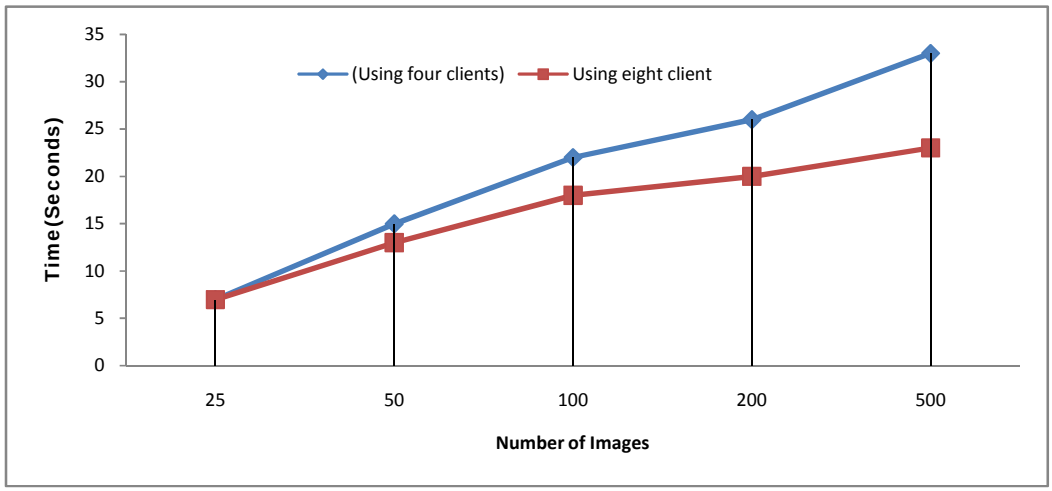

Figure 4.9: Execution Time Versus Number of Images for 4, 8 clients.

We then extend this experiment for a database with up to 3000 images, and we used 8, 12 and 16 clients .Table 4.9 gives the results of this experiment.

Table 4.9.Execution Time Versus Number of Images for 8, 12, 16 Clients.

Number of image in the data base	Time Using eight client (second)	Time Using twelve client (second)	Time Using sixteen client (second)
500	23	13	10
1000	110	90	135
1500	210	180	210
2000	400	360	300
2500	660	530	480
3000	1120	1020	840

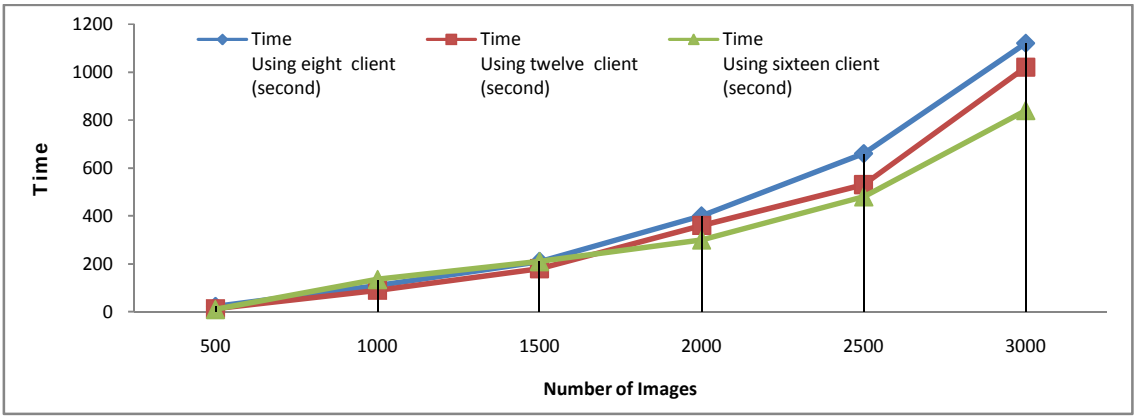

Figure 4.10: Execution Time Versus. Number of Images for 8, 12, 16 Clients.

4.6 Saving the Hash Values in the Database

To improve the efficiency of the comparison software, during the creation of the images database, we compute the hash value for each image, and save it in the database. The following experiment outlines the time required using this technique. This is like an overhead at the beginning, but it saves time during the comparison requests that come later.

Table 4.10.Time for Saving the Hash Values in Database.

Number of Image in The Database	Time spent to save the hash values in database (second)
500	25
1000	100
1500	470
2000	600
2500	723
3000	1003

4.7 Mechanism of Dividing the Work between Parallel Copies

The server administrator decides on the number of copies. Then, the server divides the images between the working copies evenly. Then, each client will start comparing each image of his part with all other images in the database. (Each image will exclude itself). The client marks only one copy of repeating files, by setting the flag field to zero for repeating images.

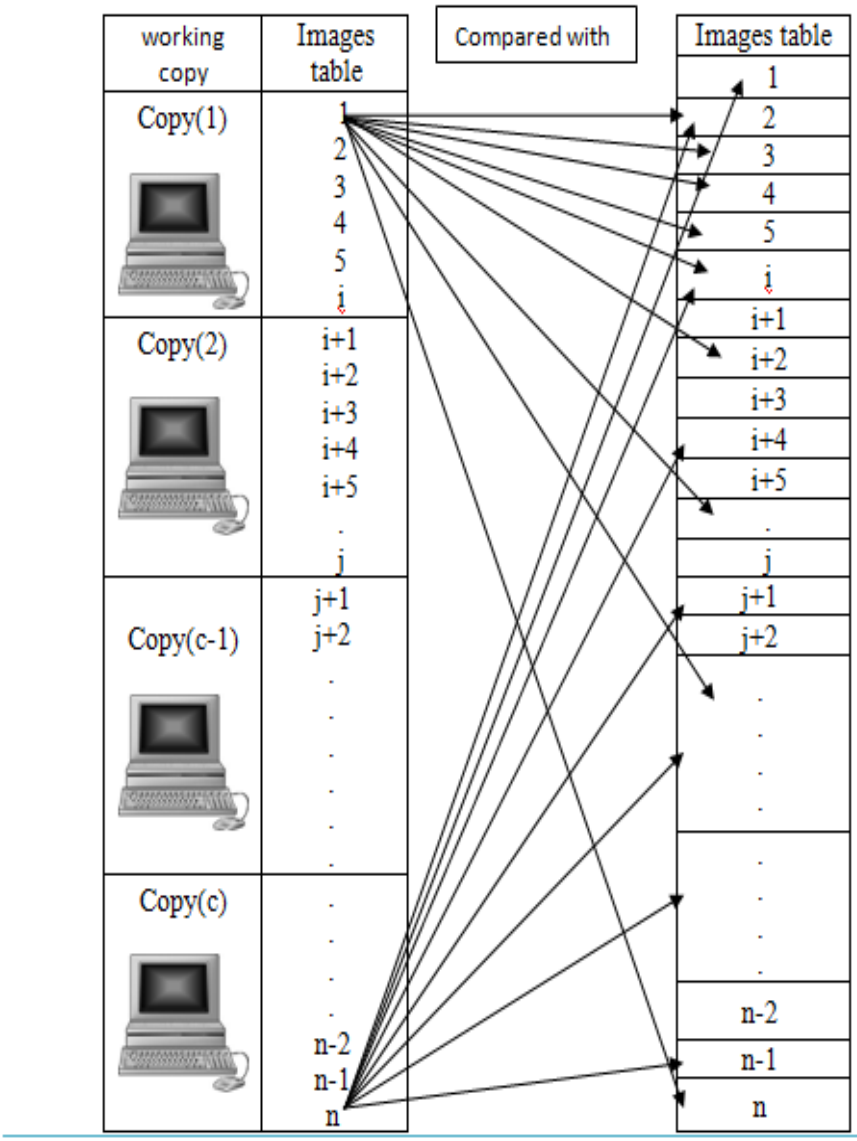

Figure 4.11: Processing of Images.

If we have (n) images in the database, using the sequential technique, the software should compare each image with (n-1) other images.

The total working time of software can be computed as follows:

T (sequential) = (image (1)*n-1+image (2)*n-1+image (3)*n-1+---+---+image (n)*n-1) (1)

T (sequential) = n * (n-1) (2)

Where n=total number of images. And i= index for each image.

(Image (1)*n-1= means the first image is compared with all other images).

On the other hand, if we use the parallel technique, the total time software works can be computed as follows:

Time for first copy = (image (1)*n-1+ image (2)*n-1+ image (3)*n-1+-----+----+--+ image (n/c)*n-1) (3)

Time for second copy = (image (n/c+1)*n-1+ image (n/c+2)*n-1+ image (n/c+1)*n-1+---+---+ image (n/c + n/c)*n-1) (4)

Therefore, the total time of parallel execution time is:

T (parallel) = n*n/c (5)

Where n=total number of images. c=total number of working copies. And i= index for each image.

Image (1)*n-1= means image (1) is compared with the other images.

It can be shown that the parallel technique is much more efficient.

Let us assume that number of images in our data base is 500.

a) With the Sequential technique:

T (sequential) == n * (n-1)=500*(500-1)=249500 Steps (comparison).

In our experiment, after running the software using 500 images in the database. It takes 145 second to finish the execution.

b) With the Parallel technique: (assuming 16 copies)

n=500. c=16.

T (parallel) =n*n/c=500*500/16=15625 Steps (comparison).

After running the software using 500 images. It takes 10 seconds to finish the execution.

If we divide the sequential time by the number of working copies, the theoretical expected parallel execution time is = 145/16=9.06. In the experiment, it takes 10 seconds to finish using 16 copies.

The reasons of this extra time are:

1) The server needs time to count the number of images the database.

2) Communication time between the server and the client's .We need time to divide the images between the clients. This is added to the time needed for running the copies.

To make the proposed method more efficient, we allow many copies of our software to work in parallel. The number of images in the database is divided evenly between the parallel copies. The database administrator decides on how many copies should be run depending on the total number of images in the database (The Hash Values is saved in the Database). The results of this experiment are listed below, in Table 4.11.

Table 4.11.Execution Time Versus. Number of Images for 4, 8, 12, 16 Clients, Using Multiple Copies of the Program.

Number of Image in The Database	Using Four Copies (sec)	Using Eight Copies (sec)	Using Twelve Copies (sec)	Using Sixteen Copies (sec)
500	5	4	3	1
1000	26	16	13	10
1500	120	90	57	25
2000	150	109	66	33
2500	180	140	100	60
3000	270	220	150	90

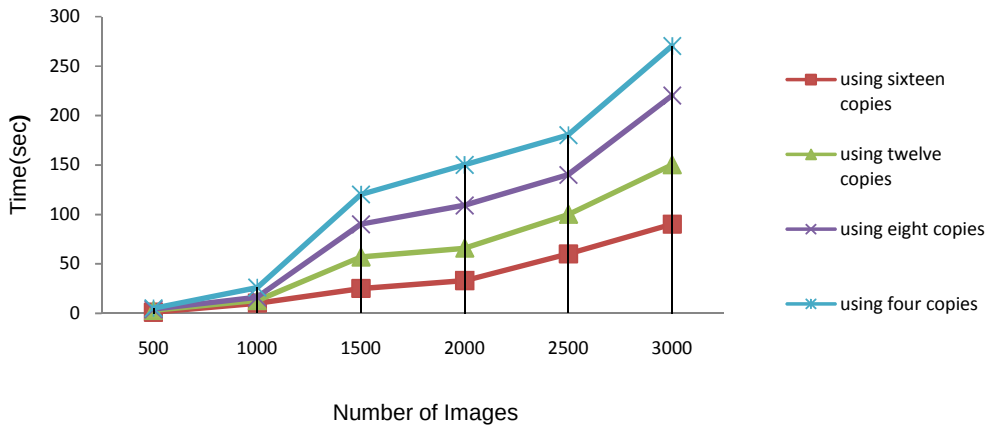

Figure 4.12: Execution Time Versus. Number of Images for 4, 8, 12, 16 Clients, Using Copies of the Program.

Figure 4.13 shows how search time is improved for a Windows 7 environment on a server PC which has core 2 duo CPU 1.83 GHz and 3.00 GB RAM. The improvement of using multiple copies is more obvious when the database has a large number of images.

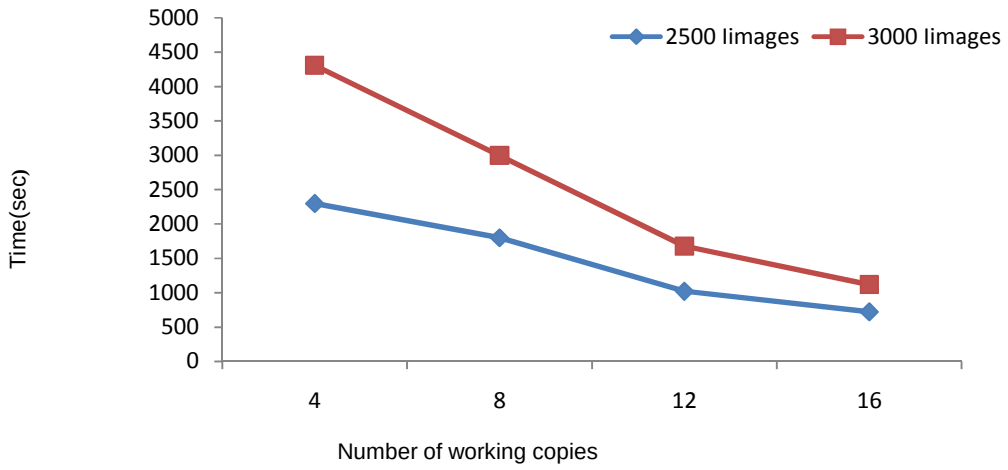

Figure 4.13: Execution Time Versus. Number of Working Copies for 2500 and 3000 Images.

4.8 Comparing dynamically way versus. Saving the Hash Values earlier in the Database

The aim of the next experiment was to see how saving hash values in the database earlier effects the time spent. Table 4.12 and Figure.4.14. Outlines the comparison between dynamic way (computing hash values when required) versus. Saving the hash values earlier in database.

Table 4.12.Dynamic Way Versus. Saving the Hash Values in Database.

Number of image in the data base	**Execution Time Using sixteen client getting the hash values in dynamic way (seconds)**	**Execution Time Using sixteen client saving the hash values in database (seconds)**
500	10	1
1000	135	10
1500	210	25
2000	300	33
2500	480	60
3000	840	90

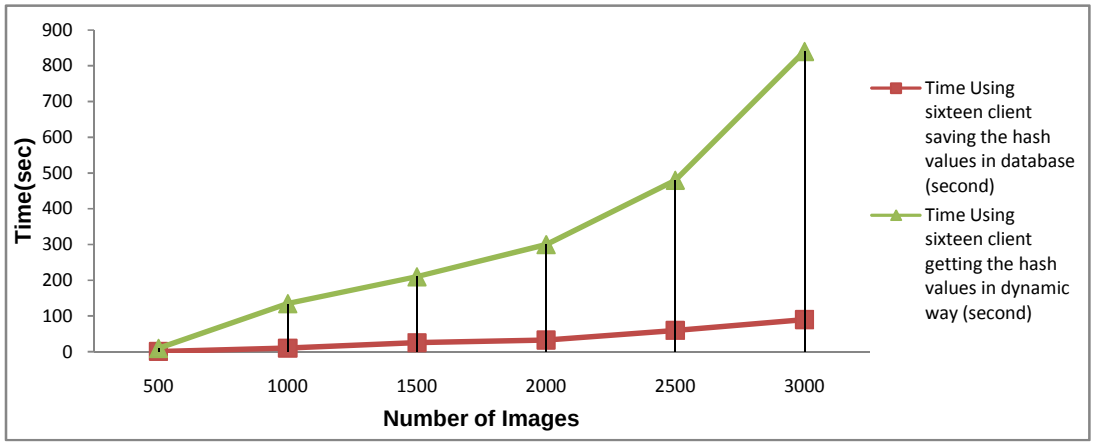

Figure 4.14: Dynamic Way Versus. Saving the Hash Values in Database.

Figure 4.14 Show that saving the hash values in database, lead to decrease the used time for comparing the images. It's clear that saving the hash values in database more efficient than the dynamic way .The improvement is more obvious when the database has a large number of images.

Chapter 5

STUDIES ON FINDING SIMILAR IMAGES

5.1 Introduction

The software developed discussed in the previous chapter is based on comparing images for exact match. Another approach is query by example, which attempt to find images similar to the one given by the user. To observe the effects of this approach, we have developed a second module which employs a query by example search technique.

In this module, a different way to get images through the Internet is proposed. The current popular way to find images is by writing a keyword in the query text box. The search engine will then try to get this image for the user. However, sometimes it is hard to explain by typing just keywords what we actually want. We may have an image and we may want to get images similar to that one. Using the query by example module, one can upload a image and find similar images on the Internet.

5.2 Query by Example Mechanism

When a user wants to find images similar to what he/she has, he/she will upload the sample image and conduct a search (Figure 2). The software will analyze this image using a specific algorithm, and then check all images in the database for similarity. After finding similar images in the database, the program displays images found and the related information.

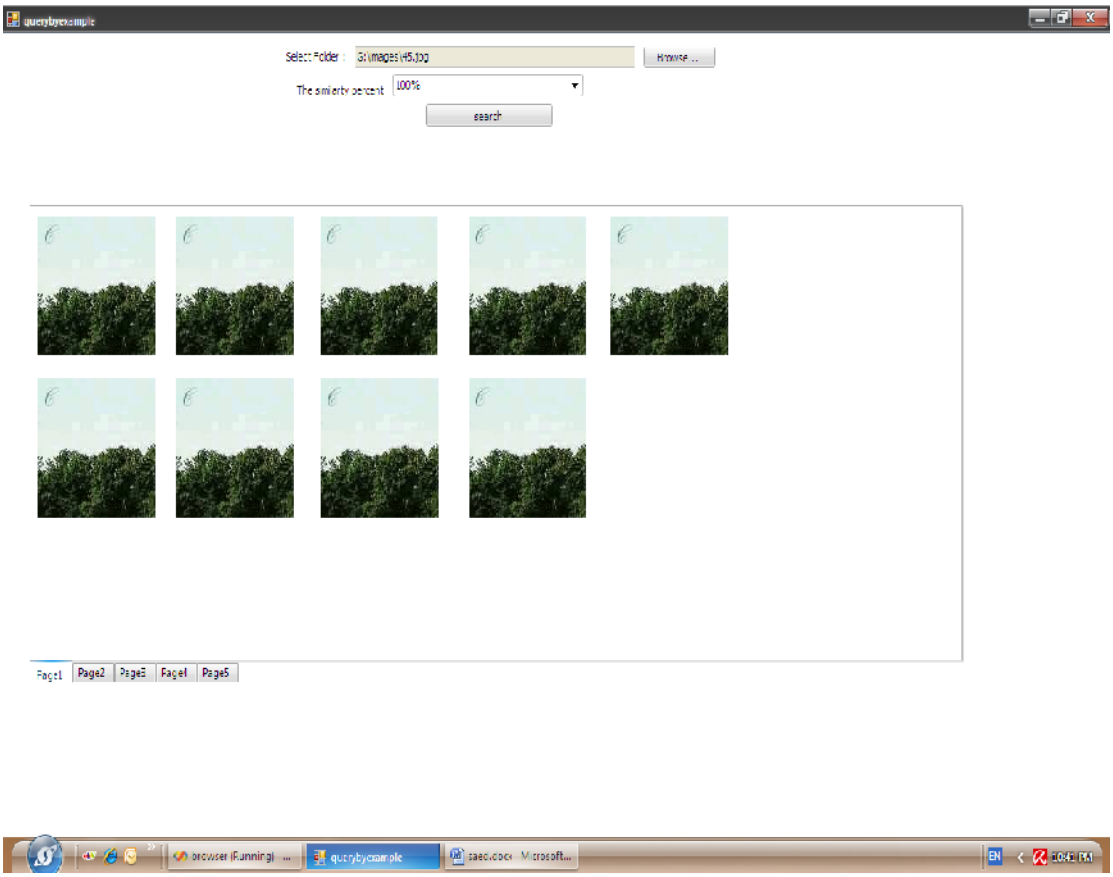

Figure 5.1: Query by Example Form.

As we know there are millions of images on the Internet, so we give the user more options to specify what he wants exactly. For this purpose, the user interface has an options entity form (Figure 5.1).

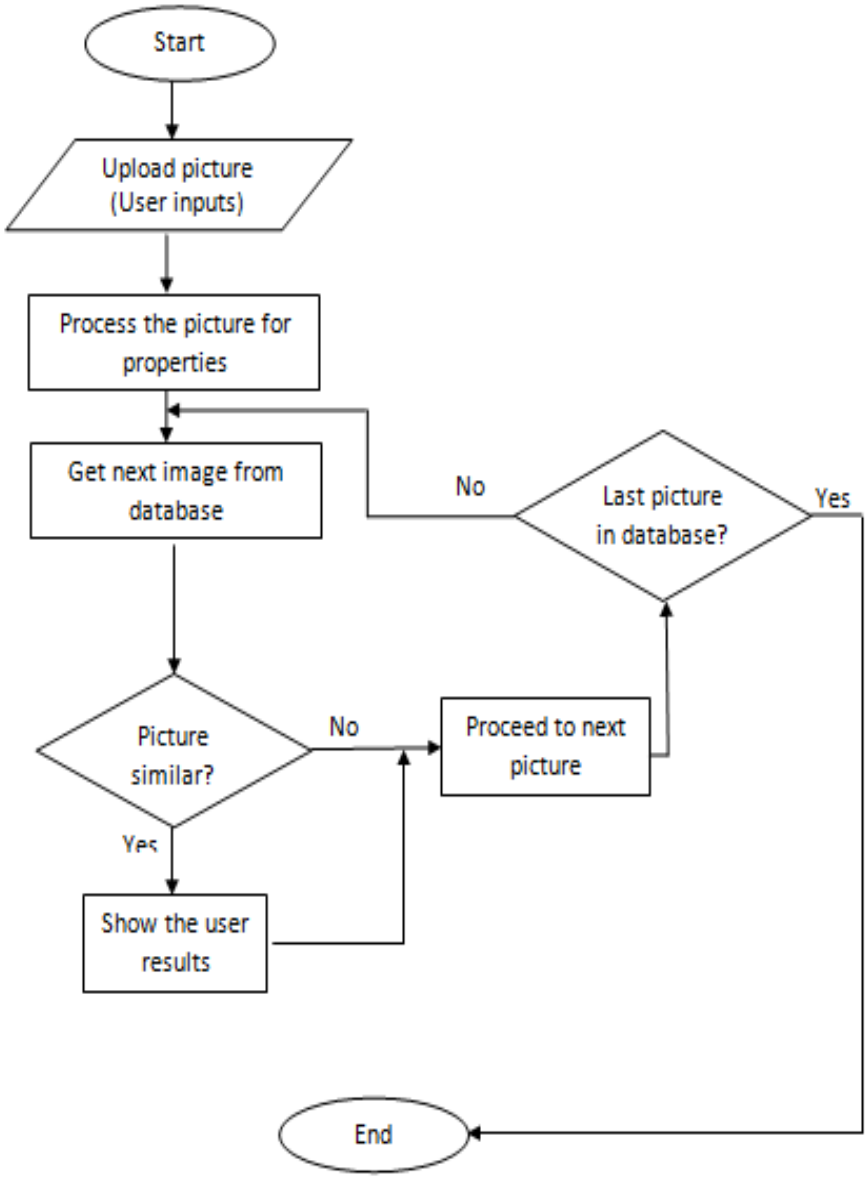

Figure 5.2: Query by Example Module Flow Chart.

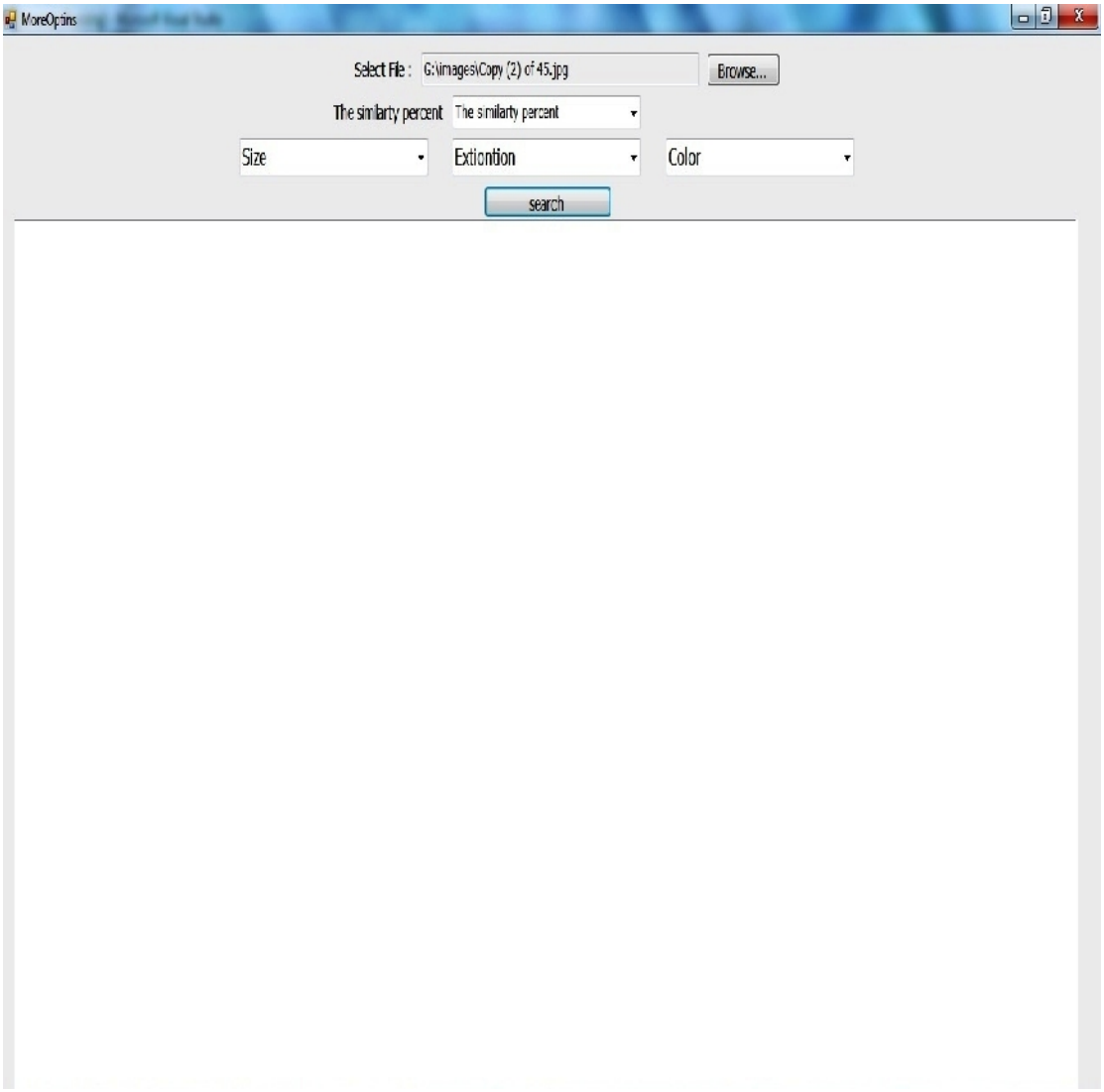

Figure 5.3: Query by Example with Options Form.

We give the user a possibility to select the size of the image if he already knows what exactly he wants. Then, the user selects the file type extension. For instance, the user selects *.ico if he wants to get icon images. Then he selects the color combination, if he wishes. All the previous options help us to get the correct images and minimize the number of searches.

The module we developed was tested using an artificial database, created by considering randomly chosen images from Internet search engines. Inside the

database we have divided the images into a group of tables depending on file extensions. For example, one table contains all images with execution (*.jpg), another one, contains all images with extensions (*.gif).

5.3 Methodology Developed For Implementing the Query by Example Techniques

Query by example software works using two different techniques to compare images. The two algorithms are:

1. Bit- wise comparison.
2. Exhaustive Template Matching.

The first technique is already discussed in Chapter Three, but we shall summarize it again, below in section 5.3.1.

5.3.1 Bit- Wise Comparison

Bit wise comparison compares all pixels one by one. Although it is an inefficient technique for comparison, its advantage is that, bit wise comparison can find certain given the percent of similarity between the compared images. After implementing this technique, we have conducted some experiments to find the efficiency of our module. Table5.1 shows us the execution time using databases with 25, 100, 200 and 500 images.

Table 5.1.Bit Wise Comparison for Similarity Using 25,100,200,500 Images.

percent of similarity	Number of images in database(25)		Number of images in database(100)		Number of images in database(200)		Number of images in database(500)	
	Time (seconds)	Number of similar images	Time (seconds)	Number of similar images	Time (seconds)	Number of similar images	Time (seconds)	Number of similar images
100%	1	1	4	1	8	1	15	5
75%	1	1	4	4	8	9	18	15
50%	1	7	5	33	8	39	19	56
25%	1	7	5	35	8	55	22	63
10%	1	9	6	55	8	78	24	111
1%	1	25	6	100	9	200	27	497

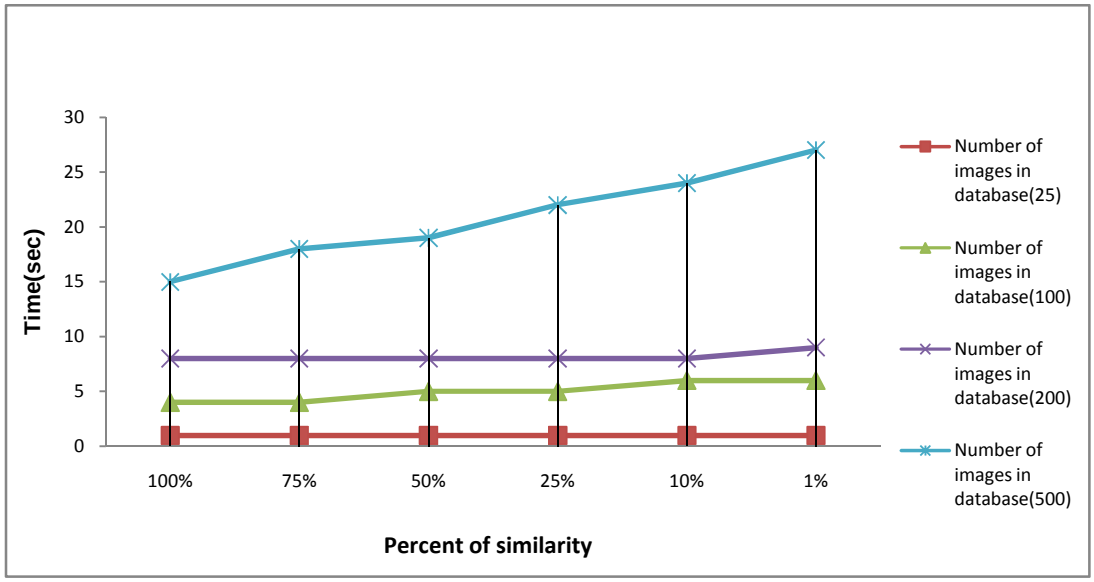

Figure 5.4: Bit Wise Comparison for Similarity Using 25,100,200,500 Images.

5.3.2 Exhaustive Template Matching

"Exhaustive template matching is a technique in digital image processing for finding small parts of an image which match a template image"[21]. The images compared must have the same size for using the exhaustive template matching

technique. Exhaustive template matching is similar to bit -wise comparison but it is more powerful. Using this technique, we can also find any percent of similarity between two images compared. Exhaustive template matching was developed as a part of AForge.NET. “AForge.NET is a framework designed for developers and researchers in the fields of Computer Vision and Artificial Intelligence - image processing, neural networks, genetic algorithms, machine learning, robotics, etc” [22]. We implemented exhaustive template matching for finding similarities between images. The following table and diagram outlines the performance of similarity comparison using exhaustive template matching.

Table 5.2.Exhaustive Template Matching Using 25,100,200,500 Images.

percent of similarity	Number of images in database(25)		Number of images in database(100)		Number of images in database(200)		Number of images in database(500)	
	Time (seconds)	Number of founded images	Time (seconds)	Number of founded images	Time (seconds)	Number of founded images	Time (seconds)	Number of founded images
100%	1	1	6	1	8	1	10	5
75%	1	1	7	8	8	15	12	15
50%	1	7	7	27	8	45	13	65
25%	1	7	7	31	9	88	13	101
10%	1	9	8	51	9	110	15	201
1%	1	25	8	100	9	200	15	350

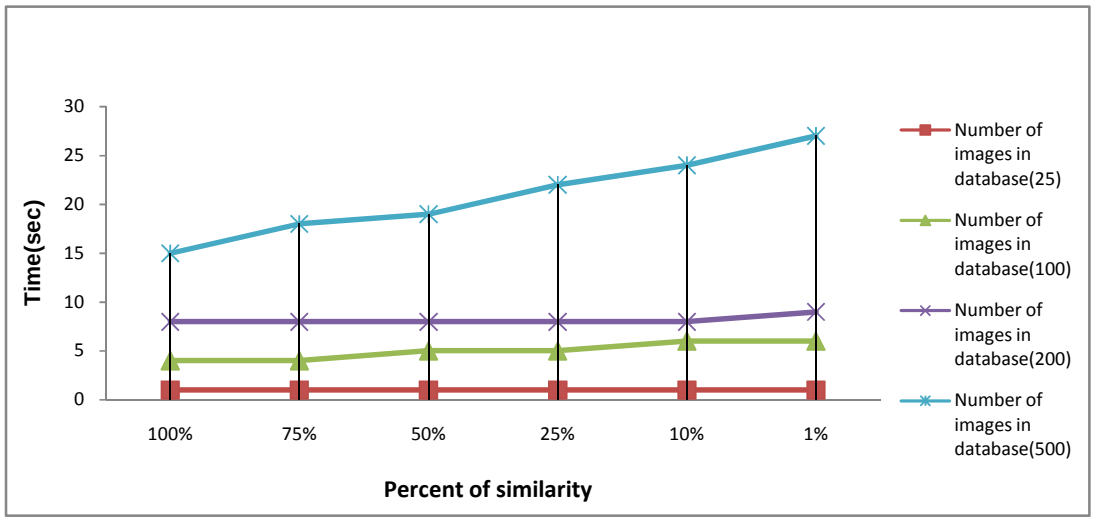

Figure 5.5: Exhaustive Template Matching Using 25,100,200,500 Images.

5.3.3 Comparison between Exhaustive Template Matching and Bit- Wise Comparison Techniques

In order to see which method is more efficient, we have compared the performance of exhaustive template matching and bit- wise comparison methods. The results are shown in table 5.3 and Figure5.6, below.

Table 5.3.Comparing Between Exhaustive Template Matching and Bit Wise Comparison.

Number of image in the data base	Time(seconds)	
	Exhaustive template matching	Bit wise comparison
25	1	1
100	5	4
200	7	8
500	10	15

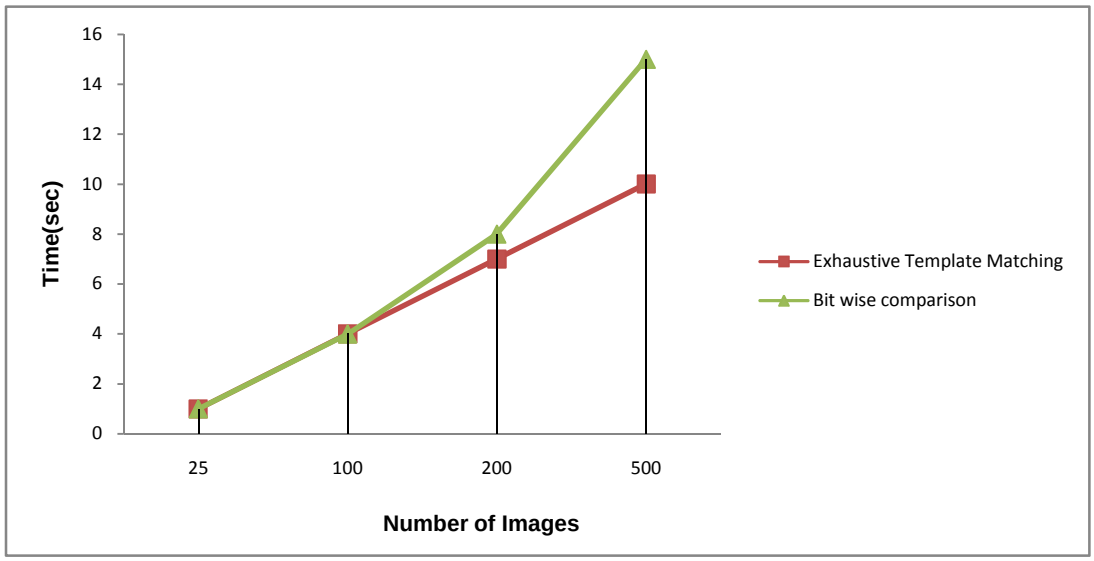

Figure 5.6: Comparing Exhaustive Template Matching and Bit Wise Comparison for finding similarity.

Chapter 6

CONCLUSION

The software developed in this work improves the efficiency of image searching by eliminating repeated occurrences of images. The output of any query will not contain repeating images, so the user does not have to go through a long list of images with repeating occurrences of the same image many times.

This software can work with any search engine. It can also work periodically on image databases. The software can create the images database. After connecting the software to the database, the software compares the hash values for the compared images, to finds repetitions, and marks only one copy of repeating files

It allows multiple copies to be run in parallel. After specify how copy of the client working. The software will divide the number of images between the working copies. Consequently, it can improve search times for images.

To make the software more efficient the administrator can make Client Interface works automatically after the administrator specify how copy of the client working. Client interface gets the required information from the database. Then, it compares the hash value with the hash values of images in the database. In this case there is no user will use the client form.

We have built two versions of our software. The first version uses parallel computers. In the first version, the software administrator will be saved and run on the server and each copy of the client software will be saved and run on different computers.

The second version uses parallel processes. In the second version the software administrator will be saved on the server and all running copy of the client software will be saved on the server.

The advantage of using the first version, the work will be divided between the working computers. In this case it is not necessary to use computers with high specifications. The disadvantage of using the first version, we need to install “VB.NET” and “SQLSERVER” on each working computer. Furthermore, the communication between the computers will lead to spend extra time, which will decrease the speed-up of the software.

The advantage of using the second version, we are using one computer as server and a number of clients at the same time. We need to install “VB.NET” and “SQLSERVER” on one computer only. Furthermore, the communication time between the working copies of the software will be less than the computation time in comparison to the first version. Hence, the peed-up is increased. The disadvantage of using the second version, we need a computer with high properties.

The software can process a very large number of images in the database. For example, the expected elapsed time to process a million of images in our database is 500 minute (8.2 hours) by using a server PC with core 2 duo CPU of 1.83 GHz frequency and 3.00 GB of RAM. In the case of using a high quality server, the elapsed time will be reduced

It is clear that, the speed-up will be increased if the number of working copy is increased. But, if we run a number of working copies greater than the required copies to compare the images in our database; the speed–up will be decreased, due to exchanging information between the server and the working copies.

We have planned to make the software multipurpose. The second module implements a query by example technique. In this module, a different way to get images through the internet is proposed .Query by example module works using three different techniques to compare the images.

Those three algorithms are: Bit wise comparison, hash comparison and exhaustive template matching. It is also possible to find images similar to the one user upload. Query by example software improves search efficiency.

Currently, the program works for the comparison of image files. We are planning to improve to use it for the audio and video files. In this case, the software will work with a multimedia database. So, the output of any multimedia query will not contain repeating files.

As we mentioned before, the second module implements a query by example technique. We are planning to improve query by example, by giving the user more option. Furthermore, we will try to use parallel way during query by example process.

Lately, content retrieval and object detection improved. We believe that using content retrieval and object detection in creating multimedia database increase the performance of search engines and makes getting wanted multimedia files easier.

Document clustering used to find similar web documents, and organize the web document. In our opinion, document clustering techniques have a good chance to improve the current mechanism.

REFERENCES

[1] Bernard J. Jansen, “Searching for digital images on the web”, Volume 3, Issue 4, Page(s): 249 - 254.

[2] Vermilyer, R , “ Intelligent User Interface Agents in Content-Based Image Retrieval ”,SoutheastCon, 2006. Proceedings of the IEEE, Publication Date: March 31 2005-April 2 2005 , Page(s): 136-142 .

[3] Watai, Y. Yamasaki, T. Aizawa, K , “View-Based Web Page Retrieval using Interactive Sketch Query”, Image Processing, 2007. ICIP 2007. IEEE International Conference on , Volume 6, Sept. 16 2007-Oct. 19 2007 Page(s): 357 - 360.

[4] Edward Y. Chang, “Web-Scale Multimedia Data Management: Challenges and Remedies ”, Image Analysis and Processing Workshops, 2007. ICIAPW 2007. 14th International Conference on 10-13 Sept. 2007 Digital Object Identifier 10.1109/ICIAPW.2007.47, Page(s):3 – 8.

[5] Mauricio Marin, Veronica Gil-Costa, and Carolina Bonacic, “ A Search Engine Index for Multimedia Content”, in 14th European Conference on Parallel and Distributed Computing, 2008, Page(s): 866-875.

[6] Amanda Spink, Dian Tjondronegoro, “Web search engine multimedia functionality”, Pergamon Press, Inc. Tarrytown, NY, USA, 2008, ISSN:0306-4573,2008.

[7] Charalampos Doulaverakis, Evangelia Nidelkou, Anastasios Gounaris, Yiannis Kompatsiaris, “A Hybrid Ontology and Content-Based Search Engine For Multimedia Retrieval ”,CiteSeerX -Scientific Literature Digital Library and Search Engine (United States), 2008.

[8] Hai Jin, Ruhan He,Zhensong Liao, Wenbing Tao, Qin Zhang , “A Flexible and Extensible Framework for Web Image Retrieval System”,Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on,19-25 Feb. 2006 Page(s):193 – 193.

[9] I. Bartolini , “A Multi-faceted Browsing Interface for Digital Photo Collections Export ”, Content-Based Multimedia Indexing, 2009. CBMI '09. Seventh International Workshop on In Content-Based Multimedia Indexing, 2009. CBMI '09. Seventh International Workshop on (2009), Page(s): 237-242.

[10] Ruhan He, Kaiming Liu, Naixue Xiong, Yong Zhu , “Garment Image Retrieval on the Web with Ubiquitous Camera-Phone”, Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference, Year of Publication: 2008 , Page(s): 1584-1589 .

[11] Doumenis, G. Papastefanos, S. Mateevitsi, V. Andritsopoulos, F. Achilleopoulos, N. Mikhalev, A.V, “Video index and search services based on content identification features” ,Broadband Multimedia Systems and Broadcasting, 2008 IEEE International Symposium on March 31 2008-April 2 2008, Page(s):1 – 4 .

[12] Abbas Cheddad, Joan Condell, Kevin Curran ,Paul McKevitt , “ A hash-based image encryption algorithm”, Optics Communications, Volume 283, Issue 6, 15 March 2010, Page(s): 879-893.

[13] William Stallings, “Cryptography and Network Security: Principles and Practice”, 3/E, Publisher: Prentice Hall Copyright: 2003, 681 pp.

[14] Yong Zhu, Naixue Xiong , Jong Hyuk Park and Ruhan He , “ A Web Image Retrieval Re-ranking Scheme with Cross-Modal Association Rules”, International Symposium on Ubiquitous Multimedia Computing, Issue 13,15 Oct. 2008, Page(s): 83 - 86.

[15] Aidong Zhang and Lei Zhu, “ Metadata Generation and Retrieval of Geographic Imagery”,National Conference for Digital Government Research,2001, Page(s):21 23.

[16] Keon Stevenson and Clement Leung, “ Comparative Evaluation of Web Image Search Engines for Multimedia Applications”, Multimedia and Expo, 2005. ICME 2005. IEEE International Conference , Issue 6-8 July 2005 , Page(s): 4.

[17] Fuat Uluç, Erkan Emirzade, Yıltan Bitirim , “The Impact of Number of Query Words on Image Search Engines”, Second International Conference on Internet and Web Applications and Services (ICIW'07), Issue 13,19 May 2007, Page(s): 50 – 50.

[18] Lin-Chih Chen, “Using a new relational concept to improve the clustering performance of search engines”, Information Processing and Management, (2010).

[19] Y.Y. Yao, “ Measuring Retrieval Effectiveness Based on User Preference of Documents”, American Society for Information Science , Volume 46, Issue 2, March 1995 , Page(s): 81–160.

[20] YOSSI RUBNER, CARLO TOMASI AND LEONIDAS J. GUIBAS, “ The Earth Mover’s Distance as a Metric for Image Retrieval”, International Journal of Computer Vision, Issue 2, Nov. 2000, Volume 40, Page(s): 2000.

[21] Template matching, “http://www.answers.com/topic/template-matching”, last visited (15/11/2010).

[22] AForge.NET Framework, “http://www.aforgenet.com/framework/features”, last visited (22/11/2010).

[23] Word Tracker, “http://www.wordtracker.com”, last visited (15/12/2010).

APPENDICES

Appendix A: The source code of the module.

```
'connect to the database
Private Sub connectdb_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
connectdb.Click
        Try
            txtConnectionString.Text = "Data
Source=EMU\SQLEXPRESS;Initial
Catalog=ImagesStore;Integrated Security=True"

            Dim CN As SqlConnection = New
SqlConnection(txtConnectionString.Text)

            'Initialize SQL adapter.
            Dim ADAP As SqlDataAdapter = New
SqlDataAdapter("Select * from ImagesStore ORDER BY
imageid", CN)

            'Initialize Dataset.
            Dim DS As DataSet = New DataSet()

            'Fill dataset with ImagesStore table.
            ADAP.Fill(DS, "ImagesStore")

            'Fill Grid with dataset.
            ' dataGridView1.DataSource =
DS.Tables("ImagesStore")
        Catch ex As Exception
            MessageBox.Show(ex.ToString())
        End Try
    End Sub
```

```
‘spicefy the images location

Private Sub cmdBrowse_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmdBrowse.Click
        FolderBrowserDialog1.ShowDialog()
        txtImagePath.Text
=FolderBrowserDialog1.SelectedPath.ToString()
    End Sub


Private Sub savepicters_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
savepicters.Click

Dim Files As String() =
Directory.GetFiles(FolderBrowserDialog1.SelectedPath.ToSt
ring())
        Dim Dirs As String() =
Directory.GetDirectories(FolderBrowserDialog1.SelectedPat
h.ToString())
        Dim Filename As String


        For Each Filename In Files


            If Filename.Contains(".jpg") Or
Filename.Contains(".gif") Or Filename.Contains(".JPG") Or
Filename.Contains(".GIF") Or Filename.Contains(".bmp")
Then

                'MessageBox.Show(Filename)
                Try
                    imageData = ReadAllBytes(Filename)
                    picturehash()
                    'Initialize SQL Server Connection
                    Dim CN As SqlConnection = New
SqlConnection(txtConnectionString.Text)
                    'Set insert query
                    Dim qry As String = "insert into
ImagesStore (OriginalPath,picturehash)
values(@OriginalPath,@picturehash)"
                    'Initialize SqlCommand object for
insert.
                    Dim SqlCom As SqlCommand = New
SqlCommand(qry, CN)
                    'We are passing Original Image Path
and Image byte data as sql parameters.
                    SqlCom.Parameters.Add(New
SqlParameter("@OriginalPath", CType(Filename, Object)))
```

```
                    'SqlCom.Parameters.Add(New
SqlParameter("@ImageData", CType(imageData, Object)))
                    SqlCom.Parameters.Add(New
SqlParameter("@picturehash", all))
                    'Open connection and execute insert
query.
If CN.State = ConnectionState.Closed Then
                        CN.Open()
End If

   SqlCom.ExecuteNonQuery()
                    If CN.State = ConnectionState.Open
Then
                        CN.Close()
                    End If
                    'Close form and return to list or
images.
                    ' Me.Close()
                Catch ex As Exception
                    MessageBox.Show(ex.ToString())
                End Try
            End If
        Next
        MessageBox.Show("pictures is added")

    End Sub



Private Sub updateserverinformation()
        Try 'serverinfo

            Dim CN As SqlConnection = New
SqlConnection(txtConnectionString.Text)

   Dim numofcomputer As Integer = InputBox("how many
client will work")
            'Set insert query
Dim qry As String = "Update  serverinfo SET numofpic=" &
i & ",numofcomputer=" & numofcomputer & " ,numforeach=" &
i / numofcomputer & " ,startnum=" & fnum & ",endnum=" &
lnum

            'Initialize SqlCommand object for insert.
            Dim SqlCom As SqlCommand = New
SqlCommand(qry, CN)
            SqlCom.Parameters.Add(New
SqlParameter("@endnum", lnum))

            'Open connection and execute insert query.
   If CN.State = ConnectionState.Closed Then
```

```
                CN.Open()
    End If

            SqlCom.ExecuteNonQuery()
            If CN.State = ConnectionState.Open Then
                CN.Close()
            End If
            'Close form and return to list or images.
            ' Me.Close()
        Catch ex As Exception
            MessageBox.Show(ex.ToString())
        End Try


'compare using hash values

Private Sub Button2_Click_1(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
md5.Click

        Dim index1, index2 As Integer
        Dim path1, path2 As String
        Dim stime As Date
        Dim etime As Date
        stime = Now.TimeOfDay.ToString
        TextBox3.Text = "start time=" + stime
        Dim bmp1, bmp2 As Bitmap
        Dim pic As Integer
        'database
        Dim CN As SqlConnection
        Dim qry As String
        Dim SqlCom As SqlCommand
        Dim picnum As Integer
        Dim dr As SqlDataReader
        Dim cn1 As SqlConnection
        Dim qry1 As String
        Dim SqlCom1 As SqlCommand
        Dim dr1 As SqlDataReader


        '====================
        Dim converter As New ImageConverter()
        Dim imgBytes1 As Byte() = New Byte(0) {}
        Dim imgBytes2 As Byte()


        Try
            CN = New
SqlConnection(txtConnectionString.Text)
```

```
            qry = "Select * from ImagesStore"

            'Initialize SqlCommand object for insert.
            SqlCom = New SqlCommand(qry, CN)

            'Initialize SQL adapter.
            CN.Open()
            dr = SqlCom.ExecuteReader
            If dr.HasRows = True Then
                While dr.Read

                    bmp1 =
System.Drawing.Image.FromFile(dr("OriginalPath"))


                    picnum = dr("ImageId")
                    index1 =
dr("OriginalPath").ToString.IndexOf(".")
                    path1 =
dr("OriginalPath").ToString.Substring(index1, 4).ToUpper

                    imgBytes1 =
DirectCast(converter.ConvertTo(bmp1,
imgBytes1.[GetType]()), Byte())
                    Try
                        cn1 = New
SqlConnection(txtConnectionString.Text)
                        qry1 = "Select * from ImagesStore
where ImageId <>" & picnum

                        SqlCom1 = New SqlCommand(qry1,
cn1)
                        'Initialize SQL adapter.

                        If cn1.State =
ConnectionState.Closed Then
                            cn1.Open()
                            dr1 = SqlCom1.ExecuteReader
                            If dr1.HasRows = True Then
                                If cn1.State =
ConnectionState.Closed Then
                                    cn1.Open()
                                End If
                                While dr1.Read



bmp2 = System.Drawing.Image.FromFile(dr1("OriginalPath"))
  pic = dr1("ImageId")
```

```
  index2 = dr1("OriginalPath").ToString.IndexOf(".")
   path2 = dr1("OriginalPath").ToString.Substring(index2,
4).ToUpper


   If bmp1.Size <> bmp2.Size Or path1 <> path2 Then
                                        TextBox1.Text =
TextBox1.Text + "     " +
"CompareResult.ciSizeMismatch".ToString
                                    Else

                                        imgBytes2 = New
Byte(0) {}

                                        'convert images
to byte array
                                        imgBytes2 =
DirectCast(converter.ConvertTo(bmp2,
imgBytes2.[GetType]()), Byte())

                                        Dim MD5csp As New
MD5CryptoServiceProvider
                                        Dim imgHash1() As
Byte = MD5csp.ComputeHash(imgBytes1)
                                        Dim imgHash2() As
Byte = MD5csp.ComputeHash(imgBytes2)

                                        'now let's
compare the hashes
                                        Dim count As
Integer = 0
                                        Dim j As Integer
= 0
                                        While count <
imgHash1.Length

                                            If
(imgHash1(count) = imgHash2(count)) Then

                                                j = j + 1
                                            Else
                                                Exit
While

                                            End If
                                            count += 1
                                        End While
```

```
                                        If j =
imgHash1.Length Then
TextBox2.Text = TextBox2.Text + "     " + "ok"


                                            Try


CN.Close()

cn1.Close()

deleteImagesFromDatabase(pic)

cn1.Open()
                                                dr1 =
SqlCom1.ExecuteReader


                                            Catch ex As
Exception

MsgBox(ex.Message)
                                            End Try


                                        Else

'MessageBox.Show("no")

                                        End If
                                    End If

                                End While
                            End If
                            If cn1.State =
ConnectionState.Open Then
                                cn1.Close()
                            End If
                            If CN.State =
ConnectionState.Closed Then
                                CN.Open()
                                dr = SqlCom.ExecuteReader
                            End If

                        End If


                    Catch ex As Exception
```

```
                        MsgBox(ex.Message)

                    End Try


                End While
            End If
            If CN.State = ConnectionState.Open Then
                CN.Close()
            End If
        Catch ex As Exception
            MsgBox(ex.Message)

        End Try


        etime = Now.TimeOfDay.ToString
        TextBox3.Text = TextBox3.Text + "end time=" +
etime

        MsgBox("ok finish")

        GetImagesFromDatabase()
    End Sub



'compare using bit way

Private Sub compare_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles compare.Click
        Dim stime As Date
        stime = Now.TimeOfDay.ToString
        TextBox3.Text = "start time=" + stime
        Dim bm1, bm2 As Bitmap
        Dim wid, hgt, wid1, hgt1 As Integer
        Dim eq_color As Color = Color.White
        Dim ne_color As Color = Color.Red
        Dim are_identical As Boolean = True
        Dim match As Integer = 0
        '*******************************
        ' Dim textmess As String = cpicturessever.Text
        '**********************
        Dim pic As Integer
        Try

            Dim CN As SqlConnection = New
SqlConnection(txtConnectionString.Text)
            Dim qry As String
            'If message = Windows.Forms.DialogResult.Yes
Then
```

```
            '    qry = "Select * from ImagesStore where
ImageId <=" & start
            'Else
            qry = "Select * from ImagesStore"
            'Form1.Close()


            'End If

            'Initialize SqlCommand object for insert.
            Dim SqlCom As SqlCommand = New
SqlCommand(qry, CN)
            Dim picnum As Integer

            'Initialize SQL adapter.
            Dim dr As SqlDataReader
            CN.Open()
            dr = SqlCom.ExecuteReader
            If dr.HasRows = True Then
                While dr.Read
                    bm1 =
System.Drawing.Image.FromFile(dr("OriginalPath"))
                    picnum = dr("ImageId")
                    wid = bm1.Width
                    hgt = bm1.Height

                    Try

                        Dim cn1 As SqlConnection = New
SqlConnection(txtConnectionString.Text)
                        Dim qry1 As String = "Select *
from ImagesStore where ImageId <>" & picnum

                        'Initialize SqlCommand object for
insert.
                        Dim SqlCom1 As SqlCommand = New
SqlCommand(qry1, cn1)
                        'Initialize SQL adapter.
                        Dim dr1 As SqlDataReader
                        If cn1.State =
ConnectionState.Closed Then
                            cn1.Open()
                            dr1 = SqlCom1.ExecuteReader
                            If dr1.HasRows = True Then
                                If cn1.State =
ConnectionState.Closed Then
                                    cn1.Open()
                                End If
                                While dr1.Read
                                    bm2 =
System.Drawing.Image.FromFile(dr1("OriginalPath"))
```

```
                                    pic = dr1("ImageId")
                                    '
MessageBox.Show(pic)
                                    Dim bm3 As New
Bitmap(bm2.Width, bm2.Width)
                                    If bm1.Size =
bm2.Size Then
                                        For x As Integer
= 0 To wid1 - 1
                                            For y As
Integer = 0 To hgt1 - 1
                                                If
bm1.GetPixel(x, y).Equals(bm2.GetPixel(x, y)) Then
                                                    match
+= 1

bm3.SetPixel(x, y, eq_color)

                                                Else

bm3.SetPixel(x, y, ne_color)

are_identical = False
                                                End If
                                            Next y
                                        Next x

                                        If are_identical
Then
                                            '
MessageBox.Show("The images are identical")
                                            Try

CN.Close()

cn1.Close()

deleteImagesFromDatabase(pic)

cn1.Open()
                                                dr1 =
SqlCom1.ExecuteReader


                                            Catch ex As
Exception
```

```
MsgBox(ex.Message)
                                                End Try

                                            Else
                                                '
MessageBox.Show("The images are different")
                                            End If
                                        Else
'MessageBox.Show("The images are different")
                                        End If
                                        '
MessageBox.Show(match)

                                        Me.Cursor =
Cursors.Default



                                        bm2.Dispose()
                                        If cn1.State =
ConnectionState.Closed Then
                                            cn1.Open()
                                        End If


                                    End While
                                End If
                                If cn1.State =
ConnectionState.Open Then
                                    cn1.Close()
                                End If
                            End If
                            bm1.Dispose()
                            If CN.State =
ConnectionState.Closed Then
                                CN.Open()
                                dr = SqlCom.ExecuteReader
                            End If
                        Catch ex As Exception
                            ' MsgBox(ex.Message)

                        End Try
                                        End While
                End If
                If CN.State = ConnectionState.Open Then
                    CN.Close()
                End If
            Catch ex As Exception
                MsgBox(ex.Message)
```

```
        End Try
        Dim etime As Date
        etime = Now.TimeOfDay.ToString
        TextBox3.Text = TextBox3.Text + "end time=" +
etime

        MsgBox("ok")

        GetImagesFromDatabase()
    End Sub



'compare using save the hash values in the database

Private Sub Button5_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button5.Click
        Dim index1, index2 As Integer
        Dim path1, path2 As String
        Dim stime As Date
        Dim etime As Date
        stime = Now.TimeOfDay.ToString
        TextBox3.Text = "start time=" + stime
        Dim bmp1, bmp2 As Bitmap
        Dim pic As Integer
        'database
        Dim CN As SqlConnection
        Dim qry As String
        Dim SqlCom As SqlCommand
        Dim picnum As Integer
        Dim dr As SqlDataReader
        Dim cn1 As SqlConnection
        Dim qry1 As String
        Dim SqlCom1 As SqlCommand
        Dim dr1 As SqlDataReader

        Try
            CN = New
SqlConnection(txtConnectionString.Text)

            qry = "Select * from ImagesStore"

            'Initialize SqlCommand object for insert.
            SqlCom = New SqlCommand(qry, CN)

            'Initialize SQL adapter.
            CN.Open()
            dr = SqlCom.ExecuteReader
            If dr.HasRows = True Then
```

```
                Dim hash As String
                While dr.Read

                    'bmp1 =
Image.FromFile(dr("OriginalPath"))

                    picnum = dr("ImageId")
                    hash = dr("picturehash")

                    Try
                        cn1 = New
SqlConnection(txtConnectionString.Text)
                        qry1 = "Select * from ImagesStore
where ImageId <>" & picnum

                        SqlCom1 = New SqlCommand(qry1,
cn1)
                        'Initialize SQL adapter.

                        If cn1.State =
ConnectionState.Closed Then
                            cn1.Open()
                            dr1 = SqlCom1.ExecuteReader
                            If dr1.HasRows = True Then
                                If cn1.State =
ConnectionState.Closed Then
                                    cn1.Open()
                                End If
                                Dim hash1 As String
                                While dr1.Read
                                    hash1 =
dr1("picturehash")
                                    pic = dr1("ImageId")
                                    If hash = hash1 Then
                                        Try

                                            CN.Close()
                                            cn1.Close()

deleteImagesFromDatabase(pic)

                                            cn1.Open()
                                            'If CN.State
= ConnectionState.Closed Then
                                            '
CN.Open()
                                            '    'dr =
SqlCom.ExecuteReader
                                            'End If
                                            dr1 =
SqlCom1.ExecuteReader
```

```
                                        Catch ex As
Exception

MsgBox(ex.Message)
                                        End Try
                                    End If

                                End While
                            End If
                            If cn1.State =
ConnectionState.Open Then
                                cn1.Close()
                            End If
                            If CN.State =
ConnectionState.Closed Then
                                CN.Open()
                                dr = SqlCom.ExecuteReader
                            End If

                        End If
                    Catch ex As Exception
                        MsgBox(ex.Message)

                    End Try

                End While
            End If
            If CN.State = ConnectionState.Open Then
                CN.Close()
            End If
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
        etime = Now.TimeOfDay.ToString
        TextBox3.Text = TextBox3.Text + "end time=" +
etime
        MsgBox("ok finish")
        GetImagesFromDatabase()
    End Sub
```

```
'compare using exhaustive template matching

Private Sub Button6_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button6.Click
        Dim index1, index2 As Integer
        Dim path1, path2 As String
        Dim stime As Date
        Dim etime As Date
        stime = Now.TimeOfDay.ToString
        TextBox3.Text = "start time=" + stime
        Dim bmp1, bmp2 As Bitmap
        Dim pic As Integer
        'database
        Dim CN As SqlConnection
        Dim qry As String
        Dim SqlCom As SqlCommand
        Dim picnum As Integer
        Dim dr As SqlDataReader
        Dim cn1 As SqlConnection
        Dim qry1 As String
        Dim SqlCom1 As SqlCommand
        Dim dr1 As SqlDataReader


        Try
            CN = New
SqlConnection(txtConnectionString.Text)

            qry = "Select * from ImagesStore"

            'Initialize SqlCommand object for insert.
            SqlCom = New SqlCommand(qry, CN)

            'Initialize SQL adapter.
            CN.Open()
            dr = SqlCom.ExecuteReader
            If dr.HasRows = True Then
                While dr.Read

                    bmp1 =
System.Drawing.Image.FromFile(dr("OriginalPath"))
                    picnum = dr("ImageId")
                    index1 =
dr("OriginalPath").ToString.IndexOf(".")
                    path1 =
dr("OriginalPath").ToString.Substring(index1, 4).ToUpper

                    Try
                        cn1 = New
SqlConnection(txtConnectionString.Text)
```

```
                        qry1 = "Select * from ImagesStore
where ImageId <>" & picnum
                        SqlCom1 = New SqlCommand(qry1,
cn1)
                        'Initialize SQL adapter.

                        If cn1.State =
ConnectionState.Closed Then
                            cn1.Open()
                            dr1 = SqlCom1.ExecuteReader
                            If dr1.HasRows = True Then
                                If cn1.State =
ConnectionState.Closed Then
                                    cn1.Open()
                                End If
                                While dr1.Read
                                    bmp2 =
System.Drawing.Image.FromFile(dr1("OriginalPath"))
                                    pic = dr1("ImageId")
                                    index2 =
dr1("OriginalPath").ToString.IndexOf(".")
                                    path2 =
dr1("OriginalPath").ToString.Substring(index2, 4).ToUpper
                                    If bmp1.Size <>
bmp2.Size Or path1 <> path2 Then
                                        TextBox1.Text =
TextBox1.Text + "     " +
"CompareResult.ciSizeMismatch".ToString
                                    Else
                                        Dim tm As New
ExhaustiveTemplateMatching(0)
                                        Dim matchings As
TemplateMatch() = tm.ProcessImage(bmp1, bmp2)

''''''''''''''''''''''
                                        Dim value As
String = matchings(0).Similarity * 100
                                        If value < 100
Then
                                            value =
Val(value.Substring(0, 2) + 1)
                                        Else
                                            value = value
                                        End If


                                        If value > 90
Then
                                            TextBox2.Text
= TextBox2.Text + "     " + "ok"
```

```
                                        Try

CN.Close()

cn1.Close()

deleteImagesFromDatabase(pic)

cn1.Open()
                                            dr1 =
SqlCom1.ExecuteReader


                                        Catch ex As
Exception

MsgBox(ex.Message)
                                        End Try


                                    Else
                                        TextBox1.Text
= TextBox1.Text + "     " + "no"

                                    End If
                                End If

                            End While
                        End If
                        If cn1.State =
ConnectionState.Open Then
                            cn1.Close()
                        End If
                        If CN.State =
ConnectionState.Closed Then
                            CN.Open()
                            dr = SqlCom.ExecuteReader
                        End If

                    End If


                Catch ex As Exception
                    MsgBox(ex.Message)

                End Try

            End While
        End If
```

```
            If CN.State = ConnectionState.Open Then
                CN.Close()
            End If
        Catch ex As Exception
            MsgBox(ex.Message)

        End Try
        etime = Now.TimeOfDay.ToString
        TextBox3.Text = TextBox3.Text + "end time=" +
etime
        MsgBox("ok finish")

        GetImagesFromDatabase()
    End Sub


'resize all images

Private Sub resizeimages()
        FolderBrowserDialog1.ShowDialog()
        Dim Files As String() =
Directory.GetFiles(FolderBrowserDialog1.SelectedPath.ToSt
ring())
        Dim Dirs As String() =
Directory.GetDirectories(FolderBrowserDialog1.SelectedPat
h.ToString())
        Dim Filename As String

        Try
            Dim image1 As New Bitmap("C:\25\1.jpg")
            For Each Filename In Files
                If Filename.Contains(".jpg") Or
Filename.Contains(".gif") Or Filename.Contains(".JPG") Or
Filename.Contains(".GIF") Or Filename.Contains(".bmp")
Then
                    Dim bm As New Bitmap(Filename)
                    Dim width As Integer =
Val(image1.Width) 'image width.
                    Dim height As Integer =
Val(image1.Height) 'image height
                    Dim thumb As New Bitmap(width,
height)

                    Dim g As Graphics =
Graphics.FromImage(thumb)
                    g.InterpolationMode =
Drawing2D.InterpolationMode.HighQualityBicubic
                    g.DrawImage(bm, New Rectangle(0, 0,
width, height), New Rectangle(0, 0, bm.Width, bm.Height),
GraphicsUnit.Pixel)
                    g.Dispose()
```

```
                    Dim Path As String = Filename
                    Path = Path.Substring(10)
                    'MessageBox.Show(Path)
                    Dim path1 As String = "C:\25\" + Path
                    'MessageBox.Show(path1)
                    thumb.Save(path1,
System.Drawing.Imaging.ImageFormat.Jpeg) 'can use any
image format
                End If
            Next
            MessageBox.Show("pictures is fixed")

        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try

    End Sub



'query by example

Private Sub savepicters_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
savepicters.Click

      Dim bmp1 As System.Drawing.Image
        Dim bmp2 As System.Drawing.Image
        Dim Tabpage As Integer = 1
        Dim count As Integer = 0
        Dim stime As Date
        Dim etime As Date
        stime = Now.TimeOfDay.ToString
        TextBox3.Text = "start time=" + stime
        Dim match As Integer = 0
        Dim mismatch As Integer = 0
        Dim pic As Integer
        Try
            bmp1 =
Image.FromFile(OpenFileDialog1.FileName.ToString())
            Try
                Dim txtConnectionString As String = "Data
Source=EMU_CMPE\SQLSERVER;Initial
Catalog=browser;Integrated Security=True"

                Dim cn1 As SqlConnection = New
SqlConnection(txtConnectionString)
                Dim qry1 As String = "Select * from
ImagesStore "

                'Initialize SqlCommand object for insert.
```

```
                Dim SqlCom1 As SqlCommand = New
SqlCommand(qry1, cn1)
                'Initialize SQL adapter.
                Dim dr1 As SqlDataReader
                If cn1.State = ConnectionState.Closed
Then
                    cn1.Open()
                    dr1 = SqlCom1.ExecuteReader
                    If dr1.HasRows = True Then
                        If cn1.State =
ConnectionState.Closed Then
                            cn1.Open()
                        End If
                        While dr1.Read
                            match = 0
                            mismatch = 0
                            bmp2 =
Image.FromFile(dr1("OriginalPath"))
                            pic = dr1("ImageId")
                            If bmp1.Size <> bmp2.Size
Then
                                TextBox1.Text =
TextBox1.Text + "CompareResult.ciSizeMismatch".ToString

'MessageBox.Show("CompareResult.ciSizeMismatch")
                            Else
                                'create instance or
System.Drawing.ImageConverter to convert
                                'each image to a byte
array
                                Dim converter As New
ImageConverter()
                                'create 2 byte arrays,
one for each image
                                Dim imgBytes1 As Byte() =
New Byte(0) {}
                                Dim imgBytes2 As Byte() =
New Byte(0) {}

                                'convert images to byte
array
                                imgBytes1 =
DirectCast(converter.ConvertTo(bmp1,
imgBytes1.[GetType]()), Byte())
                                imgBytes2 =
DirectCast(converter.ConvertTo(bmp2,
imgBytes2.[GetType]()), Byte())

'now compute a hash for each image from the byte arrays
                                Dim md5 As New
MD5CryptoServiceProvider
```

```
                                    Dim imgHash1 As Byte() =
md5.ComputeHash(imgBytes1)
                                    Dim imgHash2 As Byte() =
md5.ComputeHash(imgBytes2)
                                    Dim condution As Integer
= Val(ComboBox1.Text.Substring(0,
ComboBox1.SelectedItem.Length - 1))

                                    'now let's compare the
hashes
                                    Dim i As Integer = 0
                                    While i < imgHash1.Length
AndAlso i < imgHash2.Length
                                        If (imgHash1(i) <>
imgHash2(i)) Then
                                            mismatch =
mismatch + 1
                                            If mismatch >
(100 - condution) * imgHash1.Length / 100 Then
                                                Exit While
                                            End If


                                        Else
                                            match += 1
                                        End If

                                        If match >= condution
* imgHash1.Length / 100 Then

                                            count = count + 1
                                            If count Mod 10
<> 0 Then

Me.Controls.Item("TabControl1").Controls.Item("TabPage" &
Tabpage).Controls.Item("PictureBox" &
count).BackgroundImage =
Image.FromFile(dr1("OriginalPath"))

Me.Controls.Item("TabControl1").Controls.Item("TabPage" &
Tabpage).Controls.Item("LinkLabel" & count).Text =
dr1("OriginalPath")
                                                Exit While
                                            Else

Me.Controls.Item("TabControl1").Controls.Item("TabPage" &
Tabpage).Controls.Item("PictureBox" &
count).BackgroundImage =
Image.FromFile(dr1("OriginalPath"))

Me.Controls.Item("TabControl1").Controls.Item("TabPage" &
```

```
Tabpage).Controls.Item("LinkLabel" & count).Text =
dr1("OriginalPath")
                                                Tabpage =
Tabpage + 1
                                                Exit While
                                            End If

                                        End If

                                        bmp2.Dispose()
                                        i += 1
                                    End While
                                End If
                            End While
                            Me.Cursor = Cursors.Default
                        End If
                    End If
                    If cn1.State = ConnectionState.Open Then
                        cn1.Close()
                    End If
                    bmp1.Dispose()
                    TextBox2.Text = TextBox2.Text + "    " +
count.ToString

                Catch ex As Exception
                    MsgBox(ex.Message)
                End Try

            Catch ex As Exception
                MsgBox(ex.Message)

            End Try

            etime = Now.TimeOfDay.ToString
            TextBox3.Text = TextBox3.Text + "end time=" +
etime
        End Sub



‘client part
'saveclientinfo()
Private Sub saveclientinfo()
            Try
                Dim cn1 As SqlConnection = New
SqlConnection(txtConnectionString.Text)
                Dim qry1 As String = "Select * from " &
tablename
                'Initialize SqlCommand object for insert.
```

```
            Dim SqlCom1 As SqlCommand = New
SqlCommand(qry1, cn1)
            'Initialize SQL adapter.
            Dim dr1 As SqlDataReader
            If cn1.State = ConnectionState.Closed Then
                cn1.Open()
                dr1 = SqlCom1.ExecuteReader
                If dr1.HasRows = True Then
                    While dr1.Read
                        lnum = dr1("id")
                    End While
                    computerid = lnum + 1

                    If dr1("id") = computerid Then
                        computerid = computerid + 1

                    End If
                End If
                If cn1.State = ConnectionState.Open Then
                    cn1.Close()
                End If
            End If
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
        Try 'saving client info
            Dim crySave As String = "insert into " &
tablename & " (id,ip) values ('" & computerid & "','" &
myip.ToString & "')"

            Dim CN As SqlConnection = New
SqlConnection(txtConnectionString.Text)
            'Set insert query
            'Initialize SqlCommand object for insert.
            Dim SqlCom As SqlCommand = New
SqlCommand(crySave, CN)
            'We are passing Original Image Path and Image
byte data as sql parameters.
            SqlCom.Parameters.Add(New SqlParameter("@ip",
myip.ToString))
            'Open connection and execute insert query.
            If CN.State = ConnectionState.Closed Then
                CN.Open()
            End If
            SqlCom.ExecuteNonQuery()
            If CN.State = ConnectionState.Open Then
                CN.Close()
            End If
        Catch ex As Exception
            MessageBox.Show(ex.ToString())
        End Try
```

```
    End Sub
'readclientinfo()
Private Sub readinfo()
        Try 'read from server numof pic

            Dim cn1 As SqlConnection = New
SqlConnection(txtConnectionString.Text)

            Dim qry1 As String = "Select * from
serverinfo"
            'where ip = '" & myip.ToString & "'"
            'Initialize SqlCommand object for insert.
            Dim SqlCom1 As SqlCommand = New
SqlCommand(qry1, cn1)
            'Initialize SQL adapter.
            Dim dr1 As SqlDataReader
            If cn1.State = ConnectionState.Closed Then
                cn1.Open()
                dr1 = SqlCom1.ExecuteReader
                If dr1.HasRows = True Then
                    If cn1.State = ConnectionState.Closed
Then
                        cn1.Open()
                    End If
                    dr1.Read()
                    numofpic = dr1("numofpic")
                    numforeachone = dr1("numforeach")
                    startnum = dr1("startnum")
                    numberofcomputer =
dr1("numofcomputer")
                End If
                If cn1.State = ConnectionState.Open Then
                    cn1.Close()
                End If
            End If
        Catch ex As Exception
            MsgBox(ex.Message)

        End Try

        Try 'read from client id
            Dim cn1 As SqlConnection = New
SqlConnection(txtConnectionString.Text)

            Dim qry1 As String = "Select * from " &
tablename & " where id = '" & computerid & "'"
            Dim SqlCom1 As SqlCommand = New
SqlCommand(qry1, cn1)
            'Initialize SQL adapter.
            Dim dr1 As SqlDataReader
            If cn1.State = ConnectionState.Closed Then
```

```
                cn1.Open()
                dr1 = SqlCom1.ExecuteReader
                If dr1.HasRows = True Then
                    If cn1.State = ConnectionState.Closed
Then
                        cn1.Open()
                    End If
                    dr1.Read()
                    myid = dr1("id")
                    'MessageBox.Show(myid.ToString)
                End If
                If cn1.State = ConnectionState.Open Then
                    cn1.Close()
                End If
            End If
        Catch ex As Exception
            MsgBox(ex.Message)

        End Try


        Try 'calculate from where to start
            If myid = 1 Then
                startfrom = 1
                endwhen = numforeachone
            Else
                startfrom = endwhen + 1
                endwhen = startfrom + numforeachone
                If endwhen > numofpic Then
                    endwhen = numofpic
                End If
            End If

        Catch ex As Exception
            MsgBox(ex.Message)
        End Try

    End Sub

    Private Sub readid()
        Try
            Dim cn1 As SqlConnection = New
SqlConnection(txtConnectionString.Text)
            Dim qry1 As String = "Select * from " &
tablename

            'Initialize SqlCommand object for insert.
            Dim SqlCom1 As SqlCommand = New
SqlCommand(qry1, cn1)
            'Initialize SQL adapter.
            Dim dr1 As SqlDataReader
```

```
            If cn1.State = ConnectionState.Closed Then
                cn1.Open()
                dr1 = SqlCom1.ExecuteReader
                If dr1.HasRows = True Then
                    While dr1.Read
                        lnum = dr1("id")
                    End While
                End If
                If cn1.State = ConnectionState.Open Then
                    cn1.Close()
                End If
            End If
            computerid = lnum + 1

        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub

'updateclientinfo()

Private Sub updateclientinfo(ByVal x As String, ByVal y
As String)

        Try
            Dim CN As SqlConnection = New
SqlConnection(txtConnectionString.Text)
                        Dim qry As String = "Update " &
tablename & " SET starttime = '" & x & "'" & ",endtime
='" & y & "'" & " where id = '" & myid & "'"

            'Initialize SqlCommand object for insert.
            Dim SqlCom As SqlCommand = New
SqlCommand(qry, CN)

            'Open connection and execute insert query.
            If CN.State = ConnectionState.Closed Then
                CN.Open()
            End If

            SqlCom.ExecuteNonQuery()
            If CN.State = ConnectionState.Open Then
                CN.Close()
            End If
            'Close form and return to list or images.
            ' Me.Close()
        Catch ex As Exception
            MessageBox.Show(ex.ToString())
        End Try

    End Sub
```

```
‘start comparing
Private Sub Button5_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles compare.Click
        stime = DateTime.Now.ToLongTimeString.ToString()

        TextBox3.Text = "start time=" + stime
        Dim bm1, bm2 As Bitmap
        Dim wid, hgt, wid1, hgt1 As Integer
        Dim eq_color As Color = Color.White
        Dim ne_color As Color = Color.Red
        Dim are_identical As Boolean = True
        Dim match As Integer = 0
        '*******************************
        ' Dim textmess As String = cpicturessever.Text
        '**********************
        Dim pic As Integer
        Try
                        Dim CN As SqlConnection = New
SqlConnection(txtConnectionString.Text)
            Dim qry As String
            qry = "Select * from ImagesStore where
ImageId >=" & startfrom & "and ImageId <=" & endwhen


            'Initialize SqlCommand object for insert.
            Dim SqlCom As SqlCommand = New
SqlCommand(qry, CN)
            Dim picnum As Integer

            'Initialize SQL adapter.
            Dim dr As SqlDataReader
            CN.Open()
            dr = SqlCom.ExecuteReader
            If dr.HasRows = True Then
                While dr.Read
                     bm1 =
Image.FromFile(dr("OriginalPath"))
                   picnum = dr("ImageId")
                   wid = bm1.Width
                   hgt = bm1.Height

                   Try

                        Dim cn1 As SqlConnection = New
SqlConnection(txtConnectionString.Text)
                        Dim qry1 As String
                        qry1 = "Select * from ImagesStore
where ImageId <>" & picnum
                        'Initialize SqlCommand object for
insert.
```

```
                        Dim SqlCom1 As SqlCommand = New
SqlCommand(qry1, cn1)
                        'Initialize SQL adapter.
                        Dim dr1 As SqlDataReader
                        If cn1.State =
ConnectionState.Closed Then
                            cn1.Open()
                            dr1 = SqlCom1.ExecuteReader
                            If dr1.HasRows = True Then
                                If cn1.State =
ConnectionState.Closed Then
                                    cn1.Open()
                                End If
                                While dr1.Read
                                    bm2 =
Image.FromFile(dr1("OriginalPath"))
                                    pic = dr1("ImageId")
                                    '
MessageBox.Show(pic)
                                    wid1 = bm2.Width
                                    hgt1 = bm2.Height
                                    Dim wid3 As Integer =
Math.Min(bm1.Width, bm2.Width)
                                    Dim hgt3 As Integer =
Math.Min(bm1.Height, bm2.Height)
                                    Dim bm3 As New
Bitmap(wid3, hgt3)

                                    If bm1.Size =
bm2.Size Then
                                        For x As Integer
= 0 To wid1 - 1
                                            For y As
Integer = 0 To hgt1 - 1
                                                If
bm1.GetPixel(x, y).Equals(bm2.GetPixel(x, y)) Then
                                                    match
+= 1


bm3.SetPixel(x, y, eq_color)


                                                Else

bm3.SetPixel(x, y, ne_color)

are_identical = False
                                                End If
                                            Next y
                                        Next x
```

```
                                        If are_identical
Then
                                            '
MessageBox.Show("The images are identical")
                                            Try


CN.Close()

cn1.Close()

deleteImagesFromDatabase(pic)

cn1.Open()
                                                dr1 =
SqlCom1.ExecuteReader



                                            Catch ex As
Exception

MsgBox(ex.Message)
                                            End Try
                                        Else
                                            '
MessageBox.Show("The images are different")
                                        End If
                                    Else

'MessageBox.Show("The images are different")
                                    End If
                                    '
MessageBox.Show(match)

                                    Me.Cursor =
Cursors.Default

                                    bm2.Dispose()

                                End While
                            End If
                            If cn1.State =
ConnectionState.Open Then
                                cn1.Close()
                            End If
                        End If
                        bm1.Dispose()

                    Catch ex As Exception
```

```
                        ' MsgBox(ex.Message)

                    End Try
                                            If CN.State =
ConnectionState.Closed Then
                        CN.Open()
                        dr = SqlCom.ExecuteReader
                    End If
                End While
            End If
            If CN.State = ConnectionState.Open Then
                CN.Close()
            End If
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
        etime = DateTime.Now.ToLongTimeString.ToString()
        TextBox3.Text = TextBox3.Text + "end time=" +
etime
        updateclientinfo(stime, etime)
                'MsgBox("ok finish")

        GetImagesFromDatabase()

    End Sub
```