1. INTRODUCTION

INTRODUCTION

Human interactive systems have attracted a lot of research interest in recent years, especially for content-based image retrieval systems. Contrary to the early systems, which focused on fully automatic strategies, recent approaches have introduced human-computer interaction. In this paper, we focus on the retrieval of concepts within a large image collection. We assume that a user is looking for a set of images, the query concept, within a database. The aim is to build a fast and efficient strategy to retrieve the query concept. In content-based image retrieval (CBIR), the search may be initiated using a query as an example. The top rank similar images are then presented to the user. Then, the interactive process allows the user to refine his request as much as necessary in a relevance feedback loop. Many kinds of interaction between the user and the system have been proposed, but most of the time, user information consists of binary labels indicating whether or not the image belongs to the desired concept.

1.1 Purpose Of The Project:

Human interactive systems have attracted a lot of research interest in recent years, especially for content-based image retrieval systems. Contrary to the early systems, which focused on fully automatic strategies, recent approaches have introduced human-computer interaction. Here focus on the retrieval of concepts within a large image collection. We assume that a user is looking for a set of images, the query concept, within a database. The aim is to build a fast and efficient strategy to retrieve the query concept. In content-based image retrieval (CBIR), the search may be initiated using a query as an example. The top rank similar images are then presented to the user.

1.2 Scope of The Project:

This project is mainly designed considering the scenarios for comparing the images dynamically. Active learning methods are used to interact with the user. Although it is used for image comparison which fulfills exact requirements according to the user.
But it is not used in case of video applications. Finally, we compare our method to existing ones using real scenario on large databases.

1.3 Intended Audience and Reading Suggestions:

The intended audience for this document is the internal guides of the organization where the team has developed the project. Further modifications and reviewing will be done by the organization and deliver a final version. The final version of this document is reviewed by the Internal Guides and Head of the Department of the college.

The sequence to follow better understanding is here Purpose, Scope, Operating requirements, Advantages, Requirements etc. In the rest of the part of this SRS is mentioned with what are our product benefits, how to use this product, how the product was developed, what are the major things we have taken into the consideration all are mentioned in this rest of the part of the SRS. In the document at first we have discussed the importance of our product and functionality of the product and the software we used and all the way how we can get utilized this software is mentioned in this document.

1.4 Design and Implementation Constraints:

The software is designed in such a way that the user can easily interact with the screen because they are GUI. The customer has to enter the image dynamically and select any folder then start comparing the images one by one. So the set of similar images are displayed.

1.5 User Documentation:

In our user manual we are going perform image comparison in a very interactive way, which can be understandable by a new person who is going to use it. If a new person is using it, necessary information will be provided. We are going to explain each and every step clearly about our product so that any user can easily understand it.
2. SYSTEM ANALYSIS

2.1 Existing System:

In the existing system the CBIR method faced a lot of disadvantage in case of the image retrieval. The following are the main disadvantage faced in case of the medical field - Medical image description is an important problem in content-based medical image retrieval. Hierarchical medical image semantic features description model is proposed according to the main sources to get semantic features currently. Hence we propose the new algorithm to overcome the existing system. In existing system, Images were first annotated with text and then searched using a text-based approach from traditional database management systems.

2.2 Proposed System:

In the proposed system we use the retain technique. In this technique the user gives his input in the form of an image. We then check for the images in the training set rather than going to the database to search the images. The training set contains the images that are frequently searched by the user. Then all the relevant images are compared and the top rank similar images are displayed as the output. Here the efficiency and accuracy are increased and the drawbacks of the existing system are overcome.

2.3 Project Description:

The project entitled as “Active Learning Methods for Interactive Image Retrieval” developed using .NET using C#. Modules display as follows.

1. Comparable image
   1.1 Image Utility
      1.1.1 RGB Projections
2. Similarity measure
3. Graph
2.3.1 Comparable Image:

From a method called as Process Image in the main method we come to the comparable images module. It acts as a loop. It has one sub method in it. That method is called as Image Utility.

2.3.1.1 Image Utility

The Image Utility method senses each and the red blue and green colors in every pixel and also calculates the luminosity and adds it to the horizontal and vertical projections. It also locks the bits of the image so that other users do not have the right to modify the images being used. After done with all the calculations it unlocks the bits of the image. It has a sub method called as the RGB Projections.

2.3.1.1.1 RGB Projections

The RGB Projection senses each and every pixel in both horizontal and vertical directions. Basically it calculates the horizontal and vertical projections.

2.3.2 Similarity Measure:

The Similarity Measures module calculates the similarity between the source image and destination image. It also sorts the images in descending order of their similarity and displays it in the form of a grid.

2.3.3 Graph:

The Graph module displays the graphs for both source and destination images in the form of histograms. The graph for each image has two curves one for horizontal projection and the other for vertical projection. We even calculate the co ordinates for each curve in this module.

Input/output

The image will take the relevant image what the user search. One can see that we have selected concepts of different levels of complexities. The performances go from few
percentages of Mean average precision to 89%. The concepts that are the most difficult to retrieve are very small and/or have a very diversified visual content.

2.4 Image Retrieval:

An image retrieval system is a computer system for browsing, searching and retrieving images from a large database of digital images. Most traditional and common methods of image retrieval utilize some method of adding metadata such as captioning, keywords, or descriptions to the images so that retrieval can be performed over the annotation words. Manual image annotation is time-consuming, laborious and expensive; to address this, there has been a large amount of research done on automatic image annotation. Additionally, the increase in social web applications and the semantic web have inspired the development of several web-based image annotation tools.

Search Methods

Image search is a specialized data search used to find images. To search for images, a user may provide query terms such as keyword, image file/link, or click on some image, and the system will return images "similar" to the query. The similarity used for search criteria could be meta tags, color distribution in images, region/shape attributes, etc.

- Image meta search - search of images based on associated metadata such as keywords, text, etc.
- Content-based image retrieval (CBIR) – the application of computer vision to the image retrieval. CBIR aims at avoiding the use of textual descriptions and instead retrieves images based on their visual similarity to a user-supplied query image or user-specified image features.

2.5 CBIR:

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

"Content-based" means that the search will analyze the actual contents of the image. The term 'content' in this context might refer to colors, shapes, textures, or any other information that can be derived from the image itself. Without the ability to examine image content, searches must rely on metadata such as captions or keywords, which may be laborious or expensive to produce.

History

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

Technical Progress

There is a growing interest in CBIR because of the limitations inherent in metadata-based systems, as well as the large range of possible uses for efficient image retrieval. Textual information about images can be easily searched using existing technology, but requires humans to personally describe every image in the database. This is impractical for very large databases, or for images that are generated automatically, e.g. from surveillance cameras. It is also possible to miss images that use different synonyms in their descriptions. Systems based on categorizing images in semantic classes like "cat" as a subclass of "animal" avoid this problem but still face the same scaling issues.

Potential uses for CBIR include:

- Art collections
- Photograph archives
- Retail catalogs
• Medical diagnosis
• Crime prevention
• The military
• Intellectual property
• Architectural and engineering design
• Geographical information and remote sensing systems

CBIR Software Systems and Techniques

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

Public CBIR Systems

See CBIR engines for examples of publicly available and accessible CBIR systems.

Query Techniques

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

Query by Example

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

Options for providing example images to the system include:

• A preexisting image may be supplied by the user or chosen from a random set.
• The user draws a rough approximation of the image they are looking for, for example with blobs of color or general shapes.

This query technique removes the difficulties that can arise when trying to describe images with words.
Semantic Retrieval

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

Other Query Methods

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

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

Content Comparison Techniques

The sections below describe common methods for extracting content from images so that they can be easily compared. The methods outlined are not specific to any particular application domain.

Color

Retrieving images based on color similarity is achieved by computing a color histogram for each image that identifies the proportion of pixels within an image holding specific values (that humans express as colors). Current research is attempting to segment color proportion by region and by spatial relationship among several color regions.
Examining images based on the colors they contain is one of the most widely used techniques because it does not depend on image size or orientation. Color searches will usually involve comparing color histograms, though this is not the only technique in practice.

**Texture**

Texture measures look for visual patterns in images and how they are spatially defined. Textures are represented by Texel’s which are then placed into a number of sets, depending on how many textures are detected in the image. These sets not only define the texture, but also where in the image the texture is located. Texture is a difficult concept to represent. The identification of specific textures in an image is achieved primarily by modeling texture as a two-dimensional gray level variation. The relative brightness of pairs of pixels is computed such that degree of contrast, regularity, coarseness and directionality may be estimated (Tamura, Mori & Yamawaki, 1978). However, the problem is in identifying patterns of co-pixel variation and associating them with particular classes of textures such as “silky, or “rough.

**Shape**

Shape does not refer to the shape of an image but to the shape of a particular region that is being sought out. Shapes will often be determined first applying segmentation or edge detection to an image. Other methods like [Tushabe and Wilkinson 2008] use shape filters to identify given shapes of an image. In some case accurate shape detection will require human intervention because methods like segmentation are very difficult to completely automate.

**Applications**

Some software producers are trying to push CBIR based applications into the filtering and law enforcement markets for the purpose of identifying and censoring images with skin-tones and shapes that could indicate the presence of nudity, with controversial results.
The CBIR technology has been used in several applications such as

- fingerprint identification
- biodiversity information systems
- digital libraries
- crime prevention
- medicine
- historical research

**Medical Applications**

The use of CBIR can result in powerful services that can benefit biomedical information systems. Three large domains can instantly take advantage of CBIR techniques: teaching, research, and diagnostics. From the teaching perspective, searching tools can be used to find important cases to present to students. Research also can be enhanced by using services combining image content information with different kinds of data. For example, scientists can use mining tools to discover unusual patterns among textual (e.g., treatments reports, and patient records) and image content information. Similarity queries based on image content descriptors can also help the diagnostic process. Clinicians usually use similar cases for case-based reasoning in their clinical decision-making process. In this sense, while textual data can be used to find images of interest, visual features can be used to retrieve relevant information for a clinical case (e.g., comments, related literature, HTML pages, etc.).

**Biodiversity Information Systems**

Biologists gather many kinds of data for biodiversity studies, including spatial data, and images of living beings. Ideally, Biodiversity Information Systems (BIS) should help researchers to enhance or complete their knowledge and understanding about species and their habitats by combining textual, image content-based, and geographical queries. An example of such a query might start by providing an image as input (e.g., a photo of a fish) and then asking the system to “Retrieve all database images containing fish whose
fins are shaped like those of the fish in this photo”. A combination of this query with textual and spatial predicates would consist of “Show the drainages where the fish species with ‘large eyes’ coexists with fish whose fins are shaped like those of the fish in the photo”.

**Digital Libraries**

There are several digital libraries that support services based on image content. One example is the digital museum of butterflies, aimed at building a digital collection of Taiwanese butterflies. This digital library includes a module responsible for content-based image retrieval based on color, texture, and patterns. In a different image context, Zhu et al. present a content-based image retrieval digital library that supports geographical image retrieval. The system manages air photos which can be retrieved through texture descriptors. Place names associated with retrieved images can be displayed by cross referencing with a Geographical Name Information System (GNIS) gazetter.

In this same domain, Bergman et al. describe architecture for storage and retrieval of satellite images and video data from a collection of heterogeneous archives. Other initiatives cover different concepts of the CBIR area.

**Digital Crime Scene Images**

The digital image databases maintain large collection of images. They can be used to store, archive, browse and search interested data. Nowadays, fingerprint images have been used in law enforcement and become increasingly popular for access control to secure information and identity. The relevance of the digital image databases for crime scene applications is to organize collections of scene pictures, evidence pictures, trace evidence images, vehicle number images, identification documents and investigation documents, and so on. In CBIR, we provide a retrieval method for digital image databases of crime scene photos.
2.6 SDLC Model:

Life Cycle Model

The life cycle model, which we use in our application, is RAD (Rapid Application Development) model. RAD is a concept that products can be developed faster with higher quality through,

- Gathering requirements using workshops or focus groups
- Prototyping and early, reiterative user testing of designs
- The re-use of software components
- A rigidly paced schedule that defers design improvements to the next product version.
- Less formality in reviews and other team communication.
3. SYSTEM STUDY

3.1 Feasibility Study:

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

3.2 Economical Feasibility:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.3 Technical Feasibility:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.
3.4 Social Feasibility:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.
4. SYSTEM SPECIFICATION

4.1 Hardware Requirements:

- SYSTEM : Pentium IV 2.4 GHz
- HARD DISK : 40 GB
- FLOPPY DRIVE : 1.44 MB
- MONITOR : 15 VGA colour
- MOUSE : Logitech.
- RAM : 256 MB
- KEYBOARD : 110 keys enhanced.

4.2 Software Requirements:

- Operating system : Windows XP Professional
- Coding Language : C# 2.0
5. NON FUNCTIONAL REQUIREMENTS

Performance Requirements

Usability

A very interactive GUI is provided which helps the user to easily interact with the system. This application allows users to operate with little or no learning.

Portability

This application is developed on dot net platform, which runs under CLR (Common Language Runtime) of that particular operating system like Windows, UNIX, and Linux. At present we have CLR working on windows platform in the market. So this application is portable to work under CLR environment imported on windows OS.

Response Time

It is the time a generic system or functional unit takes to react to a given input. Responsiveness, how quickly an interactive system responds to user input. This will take a second or two to respond for the files with more than 100 classes.

Size

For better usage of RAM chip memory in our application constraints are followed like re-useage of the variables and objects, garbage collection of unused objects etc.

Safety Requirements

No harm is expected from the use of the product either to the OS or any data that resides on the client system.
6. SOFTWARE DESIGN

6.1 UML DESIGN:

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

The Nine UML diagrams are:

Use case diagram:

The use case diagram is used to identify the primary elements and processes that form the system. The primary elements are termed as "actors" and the processes are called "use cases." The use case diagram shows which actors interact with each use case.

Class diagram:

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelate classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.
Object diagram:

The object diagram is a special kind of class diagram. An object is an instance of a class. This essentially means that an object represents the state of a class at a given point of time while the system is running. The object diagram captures the state of different classes in the system and their relationships or associations at a given point of time.

State diagram:

A state diagram, as the name suggests, represents the different states that objects in the system undergo during their life cycle. Objects in the system change states in response to events. In addition to this, a state diagram also captures the transition of the object's state from an initial state to a final state in response to events affecting the system.

Activity diagram:

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

Sequence diagram:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

Collaboration diagram:

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.
Component diagram:

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components from part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.

Deployment diagram:

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.

6.1.1 Use case Diagram:
6.1.2 Class Diagram:

![Class Diagram Image]

6.1.3 Sequence Diagram:

![Sequence Diagram Image]
6.1.4 Collaboration Diagram:

```
check
1: the availability
2: Not available

3: find the Image
4: Compare

get average value

5: draw the graph

Graph
```

6.1.5 Activity Diagram:

```
ROE Projections

Check image

availability

new Image

Compare&Utility

Similarity Images

Average and graph values
```
6.1.6 Component Diagram:
7. SYSTEM DESIGN

7.1 Module Diagram:

- RGB Projections
- Image Utility and compare
- Computing Similarity Measure
- Average and graph values

7.2 Data Flow Diagram:

- Check Image
- RGB Projections
- New Image
- Compare& Utility
- Similarity Images
- Draw the Graph
- Exit
7.3 Project Flow Diagram:

RGB Projections → ImageUtility → Compare the Image

Find the average and draw the graph → Similarity Images

7.4 System Architecture:
7.5 E-R DIAGRAM:
8. LANGUAGE SPECIFICATION

The .NET Framework is an integral Windows component that supports building and running the next generation of applications and XML Web services. The .NET Framework is designed to fulfill the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The .NET Framework has two main components:

- The common language runtime and the .NET Framework class library.
- The common language runtime is the foundation of the .NET Framework.

You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that promote security and robustness. In fact, the concept of code management is a fundamental principle of the runtime. Code that targets the runtime is known as managed code, while code that does not target the runtime is known as unmanaged code. The class library, the other main component of the .NET Framework, is a comprehensive, object-oriented collection of reusable types that you can use to develop applications ranging from traditional command-line or graphical user interface (GUI) applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services.
The .NET Framework can be hosted by unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, thereby creating a software environment that can exploit both managed and unmanaged features. The .NET Framework not only provides several runtime hosts, but also supports the development of third-party runtime hosts.

For example, ASP.NET hosts the runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the runtime to enable ASP.NET applications and XML Web services, both of which are discussed later in this topic.

Internet Explorer is an example of an unmanaged application that hosts the runtime (in the form of a MIME type extension). Using Internet Explorer to host the runtime enables you to embed managed components or Windows Forms controls in HTML documents. Hosting the runtime in this way makes managed mobile code (similar to Microsoft® ActiveX® controls) possible, but with significant improvements that only managed code can offer, such as semi-trusted execution and isolated file storage.

The following illustration shows the relationship of the common language runtime and the class library to your applications and to the overall system. The illustration also shows how managed code operates within a larger architecture.
8.1 Features of .NET:

Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There’s no language barrier with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that components created in different languages can easily interoperate.

“.NET” is also the collective name given to various software components built upon the .NET platform. These will be both products (Visual Studio.NET and Windows.NET Server, for instance) and services (like Passport, .NET My Services, and so on).

The .NET Framework

The .NET Framework has two main parts:

1. The Common Language Runtime (CLR).

2. A hierarchical set of class libraries.

The CLR is described as the “execution engine” of .NET. It provides the environment within which programs run. The most important features are:

- Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on.
- Memory management, notably including garbage collection.
- Checking and enforcing security restrictions on the running code.
- Loading and executing programs, with version control and other such features.

The following features of the .NET framework are also worth description:
Managed Code:

The code that targets .NET, and which contains certain extra information - "metadata" - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the CLR to guarantee, for instance, safe execution and interoperability.

Managed Data:

With Managed Code comes Managed Data. CLR provides memory allocation and deallocation facilities, and garbage collection. Some .NET languages use Managed Data by default, such as C#, Visual Basic.NET and JScript.NET, whereas others, namely C++, do not. Targeting CLR can, depending on the language you’re using, impose certain constraints on the features available. As with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications - data that doesn’t get garbage collected but instead is looked after by unmanaged code.

Common Type System:

The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other, by describing types in a common way. CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception handling. As well as ensuring that types are only used in appropriate ways, the runtime also ensures that code doesn’t attempt to access memory that hasn’t been allocated to it.

Common Language Specification:

The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.
The Class Library:

.NET provides a single-rooted hierarchy of classes, containing over 7000 types. The root of the namespace is called System; this contains basic types like Byte, Double, Boolean, and String, as well as Object. All objects derive from System. Object. As well as objects, there are value types. Value types can be allocated on the stack, which can provide useful flexibility. There are also efficient means of converting value types to object types if and when necessary.

The set of classes is pretty comprehensive, providing collections, file, screen, and network I/O, threading, and so on, as well as XML and database connectivity.

The class library is subdivided into a number of sets (or namespaces), each providing distinct areas of functionality, with dependencies between the namespaces kept to a minimum.

Languages Supported By .NET:

The multi-language capability of the .NET Framework and Visual Studio .NET enables developers to use their existing programming skills to build all types of applications and XML Web services. The .NET framework supports new versions of Microsoft’s old favorites Visual Basic and C++ (as VB.NET and Managed C++), but there are also a number of new additions to the family.

Visual Basic .NET has been updated to include many new and improved language features that make it a powerful object-oriented programming language. These features include inheritance, interfaces, and overloading, among others. Visual Basic also now supports structured exception handling, custom attributes and also supports multi-threading.

Visual Basic .NET is also CLS compliant, which means that any CLS-compliant language can use the classes, objects, and components you create in Visual Basic .NET.

Managed Extensions for C++ and attributed programming are just some of the enhancements made to the C++ language. Managed Extensions simplify the task of migrating existing C++ applications to the new .NET Framework. C# is Microsoft’s new language. It’s a C-style language that is essentially “C++ for Rapid Application Development”. Unlike other languages, its specification is just the grammar of the
language. It has no standard library of its own, and instead has been designed with the intention of using the .NET libraries as its own.

Microsoft Visual J# .NET provides the easiest transition for Java-language developers into the world of XML Web Services and dramatically improves the interoperability of Java-language programs with existing software written in a variety of other programming languages.

Active State has created Visual Perl and Visual Python, which enable .NET-aware applications to be built in either Perl or Python. Both products can be integrated into the Visual Studio .NET environment. Visual Perl includes support for Active State’s Perl Dev Kit.

Other languages for which .NET compilers are available include:

- FORTRAN
- COBOL
- Eiffel

---

<table>
<thead>
<tr>
<th>ASP.NET</th>
<th>XML WEB SERVICES</th>
<th>Windows Forms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**.Net Framework**

**8.2 Features of C#.NET:**

C#.NET is also compliant with CLS (Common Language Specification) and supports structured exception handling. CLS is set of rules and constructs that are supported by the CLR (Common Language Runtime). CLR is the runtime environment
provided by the .NET Framework; it manages the execution of the code and also makes the development process easier by providing services.

C#.NET is a CLS-compliant language. Any objects, classes, or components that created in C#.NET can be used in any other CLS-compliant language. In addition, we can use objects, classes, and components created in other CLS-compliant languages in C#.NET. The use of CLS ensures complete interoperability among applications, regardless of the languages used to create the application.

**Constructors and Destructors:**

Constructors are used to initialize objects, whereas destructors are used to destroy them. In other words, destructors are used to release the resources allocated to the object. In C#.NET the sub finalize procedure is available. The sub finalize procedure is used to complete the tasks that must be performed when an object is destroyed. The sub finalize procedure is called automatically when an object is destroyed. In addition, the sub finalize procedure can be called only from the class it belongs to or from derived classes.

**Garbage Collection:**

Garbage Collection is another new feature in C#.NET. The .NET Framework monitors allocated resources, such as objects and variables. In addition, the .NET Framework automatically releases memory for reuse by destroying objects that are no longer in use.

In C#.NET, the garbage collector checks for the objects that are not currently in use by applications. When the garbage collector comes across an object that is marked for garbage collection, it releases the memory occupied by the object.

**Overloading:**

Overloading is another feature in C#. Overloading enables us to define multiple procedures with the same name, where each procedure has a different set of arguments. Besides using overloading for procedures, we can use it for constructors and properties in a class.
**Multithreading:**

C#.NET also supports multithreading. An application that supports multithreading can handle multiple tasks simultaneously, we can use multithreading to decrease the time taken by an application to respond to user interaction.

**Structured Exception Handling:**

C#.NET supports structured handling, which enables us to detect and remove errors at runtime. In C#.NET, we need to use Try…Catch…Finally statements to create exception handlers. Using Try…Catch…Finally statements, we can create robust and effective exception handlers to improve the performance of our application.

**The .Net Framework:**

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet.

**Objectives of .Net Framework:**

1. To provide a consistent object-oriented programming environment whether object codes is stored and executed locally on Internet-distributed, or executed remotely.
2. To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code.
3. Eliminates the performance problems.

There are different types of application, such as

- Windows-based applications
- Web-based applications.
9. SYSTEM TESTING AND MAINTENANCE

9.1 Testing Introduction:

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. In the testing process we test the actual system in an organization and gather errors from the new system operates in full efficiency as stated. System testing is the stage of implementation, which is aimed to ensuring that the system works accurately and efficiently.

In the testing process we test the actual system in an organization and gather errors from the new system and take initiatives to correct the same. All the front-end and back-end connectivity are tested to be sure that the new system operates in full efficiency as stated. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently.

The main objective of testing is to uncover errors from the system. For the uncovering process we have to give proper input data to the system. So we should have more conscious to give input data. It is important to give correct inputs to efficient testing.

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus the system testing is a confirmation that all is correct and an opportunity to show the user that the system works.

Inadequate testing or non-testing leads to errors that may appear few months later. This will create two problems

1. Time delay between the cause and appearance of the problem.
2. The effect of the system errors on files and records within the system.

The purpose of the system testing is to consider all the likely variations to which it will be suggested and push the system to its limits.

The testing process focuses on logical intervals of the software ensuring that all the statements have been tested and on the function intervals (i.e.,) conducting tests to
uncover errors and ensure that defined inputs will produce actual results that agree with the required results. Testing has to be done using the two common steps Unit testing and Integration testing. In the project system testing is made as follows:

The procedure level testing is made first. By giving improper inputs, the errors occurred are noted and eliminated. This is the final step in system life cycle. Here we implement the tested error-free system into real-life environment and make necessary changes, which runs in an online fashion. Here system maintenance is done every months or year based on company policies, and is checked for errors like runtime errors, long run errors and other maintenances like table verification and reports.

Scope

- IE window opens automatically on setting the resolution of the window, thus avoiding the manual work.
- Address Bar, Menu Bar, Tool Bar can be managed according to programmer’s purpose.
- Source Code can be viewed directly.
- Source code of websites whose “VIEW SOURCE CODE” option is disabled can be viewed.

Defects and failures

Not all software defects are caused by coding errors. One common source of expensive defects is caused by requirements gaps, e.g., unrecognized requirements that result in errors of omission by the program designer. A common source of requirements gaps is non-functional requirements such as testability, scalability, maintainability, usability, performance, and security. Software faults occur through the following process. A programmer makes an error (mistake), which results in a defect (fault, bug) in the software source code. If this defect is executed, in certain situations the system will produce wrong results, causing a failure. Not all defects will necessarily result in failures. For example, defects in dead code will never result in failures. A defect can turn into a failure when the environment is changed. Examples of these changes in environment include the software being run on a new hardware platform, alterations in source data or
interacting with different software. A single defect may result in a wide range of failure symptoms.

**Compatibility**

A frequent cause of software failure is compatibility with another application, a new operating system, or, increasingly, web browser version. In the case of lack of backward compatibility, this can occur (for example...) because the programmers have only considered coding their programs for, or testing the software upon, "the latest version of" this-or-that operating system. The unintended consequence of this fact is that: their latest work might not be fully compatible with earlier mixtures of software/hardware, or it might not be fully compatible with *another* important operating system. In any case, these differences, whatever they might be, may have resulted in (unintended...) software failures, as witnessed by some significant population of computer users.

**Input combinations and preconditions**

A very fundamental problem with software testing is that testing under *all* combinations of inputs and preconditions (initial state) is not feasible, even with a simple product. This means that the number of defects in a software product can be very large and defects that occur infrequently are difficult to find in testing. More significantly, non-functional dimensions of quality (how it is supposed to *be* versus what it is supposed to *do*) -- for example, usability, scalability, performance, compatibility, reliability -- can be highly subjective; something that constitutes sufficient value to one person may be intolerable to another.

**Static vs. Dynamic Testing**

There are many approaches to software testing. Reviews, walkthroughs or inspections are considered as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. The former can be, (and unfortunately in practice often is...) omitted, whereas the latter takes place when programs begin to be used for the first time - which is normally considered the beginning
of the testing stage. This may actually begin before the program is 100% complete in order to test particular sections of code (modules or discrete functions). For example, Spreadsheet programs are, by their very nature, tested to a large extent "on the fly" during the build process as the result of some calculation or text manipulation is shown interactively immediately after each formula is entered.

**Software Quality Assurance (SQA)**

Though controversial, software testing may be viewed as an important part of the software quality assurance (SQA) process. In SQA, software process specialists and auditors take a broader view on software and its development. They examine and change the software engineering process itself to reduce the amount of faults that end up in the delivered software: the so-called *defect rate*.

What constitutes an "acceptable defect rate" depends on the nature of the software. For example, an arcade video game designed to *simulate* flying an airplane would presumably have a much higher tolerance for defects than mission critical software such as that used to control the functions of an airliner that *really is* flying!

Although there are close links with SQA, testing departments often exist independently, and there may be no SQA function in some companies.

Software Testing is a task intended to detect defects in software by contrasting a computer program's expected results with its actual results for a given set of inputs. By contrast, QA (Quality Assurance) is the implementation of policies and procedures intended to prevent defects from occurring in the first place.

**Testing Methods**

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.
9.2 Black Box Testing:

Black box testing treats the software as a "black box," without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

Specification-Based Testing

Specification-based testing aims to test the functionality of software according to the applicable requirements. Thus, the tester inputs data into, and only sees the output from, the test object. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Specification-based testing is necessary, but it is insufficient to guard against certain risks.

Advantages and Disadvantages

The black box tester has no "bonds" with the code, and a tester's perception is very simple: a code must have bugs. Using the principle, "Ask and you shall receive," black box testers find bugs where programmers don't. But, on the other hand, black box testing has been said to be "like a walk in a dark labyrinth without a flashlight," because the tester doesn't know how the software being tested was actually constructed. That's why there are situations when (1) a black box tester writes many test cases to check something that can be tested by only one test case, and/or (2) some parts of the back end are not tested at all.

Therefore, black box testing has the advantage of "an unaffiliated opinion," on the one hand, and the disadvantage of "blind exploring," on the other.

9.3 White Box Testing:

White box testing, by contrast to black box testing, is when the tester has access to the internal data structures and algorithms (and the code that implement these)
Types of White Box Testing

The following types of White Box Testing exist,

- **api testing** - Testing of the application using Public and Private APIs.
- **Code coverage** - creating tests to satisfy some criteria of code coverage. For example, the test designer can create tests to cause all statements in the program to be executed at least once.
- **Fault injection methods.**
- **Mutation testing methods.**
- **Static testing** - White box testing includes all static testing.

Code Completeness Evaluation

White box testing methods can also be used to evaluate the completeness of a test suite that was created with black box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested.

Two common forms of code coverage are:

- **function coverage**, which reports on functions executed
- **and statement coverage**, which reports on the number of lines executed to complete the test.

They both return coverage metric, measured as a percentage.

9.4 Grey Box Testing:

In recent years the term grey box testing has come into common usage. This involves having access to internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level. Manipulating input data and formatting output do not qualify as "grey-box," because the input and output are clearly outside of the "black-box" that we are calling "the software under test." (This distinction is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test.) Grey box testing may also include reverse engineering to determine, for instance, boundary values or error messages.
Acceptance Testing

Acceptance testing can mean one of two things:

1. A smoke test is used as an acceptance test prior to introducing a build to the main testing process.
2. Acceptance testing performed by the customer is known as user acceptance testing (UAT).

Regression Testing

Regression testing is any type of software testing that seeks to uncover software regressions. Such regressions occur whenever software functionality that was previously working correctly stops working as intended. Typically regressions occur as an unintended consequence of program changes. Common methods of regression testing include re-running previously run tests and checking whether previously fixed faults have re-emerged.

Non Functional Software Testing

Special methods exist to test non-functional aspects of software.

- Performance testing checks to see if the software can handle large quantities of data or users. This is generally referred to as software scalability. This activity of Non Functional Software Testing is often times referred to as Load Testing.
- Usability testing is needed to check if the user interface is easy to use and understand.
- Security testing is essential for software which processes confidential data and to prevent system intrusion by hackers.
- Internationalization and localization is needed to test these aspects of software, for which a pseudo localization method can be used.

In contrast to functional testing, which establishes the correct operation of the software (correct in that it matches the expected behavior defined in the design requirements) non-functional testing verifies that the software functions properly even when it receives invalid or unexpected inputs. Software fault injection, in the form of
fuzzing, is an example of non-functional testing. Non-functional testing, especially for software, is designed to establish whether the device under test can tolerate invalid or unexpected inputs, thereby establishing the robustness of input validation routines as well as error-handling routines. Various commercial non-functional testing tools are linked from the Software fault injection page; there are also numerous open-source and free software tools available that perform non-functional testing.

9.5 Testing Process:

A common practice of software testing is performed by an independent group of testers after the functionality is developed before it is shipped to the customer. This practice often results in the testing phase being used as project buffer to compensate for project delays, thereby compromising the time devoted to testing. Another practice is to start software testing at the same moment the project starts and it is a continuous process until the project finishes.

In counterpoint, some emerging software disciplines such as extreme programming and the agile software development movement, adhere to a "test-driven software development" model. In this process, unit tests are written first, by the software engineers (often with pair programming in the extreme programming methodology). Of course these tests fail initially; as they are expected to. Then as code is written it passes incrementally larger portions of the test suites. The test suites are continuously updated as new failure conditions and corner cases are discovered, and they are integrated with any regression tests that are developed. Unit tests are maintained along with the rest of the software source code and generally integrated into the build process (with inherently interactive tests being relegated to a partially manual build acceptance process).

Testing can be done on the following levels:

- Unit testing tests the minimal software component, or module. Each unit (basic component) of the software is tested to verify that the detailed design for the unit has been correctly implemented. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.
• Integration testing exposes defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

• System testing tests a completely integrated system to verify that it meets its requirements.

• System integration testing verifies that a system is integrated to any external or third party systems defined in the system requirements.

Before shipping the final version of software, alpha and beta testing are often done additionally:

• **Alpha testing** is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

• **Beta testing** comes after alpha testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

Finally, acceptance testing can be conducted by the end-user, customer, or client to validate whether or not to accept the product. Acceptance testing may be performed as part of the hand-off process between any two phases of development.

**Regression Testing**

After modifying software, either for a change in functionality or to fix defects, a regression test re-runs previously passing tests on the modified software to ensure that the modifications haven't unintentionally caused a *regression* of previous functionality. Regression testing can be performed at any or all of the above test levels. These regression tests are often automated.
More specific forms of regression testing are known as sanity testing, when quickly checking for bizarre behavior, and smoke testing when testing for basic functionality.

**Measuring Software Testing**

Usually quality is constrained to such topics as correctness, completeness, security, but can also include more technical requirements as described under the ISO standard ISO 9126, such as capability, reliability, efficiency, portability, maintainability, compatibility and usability.

There are a number of common software measures, often called "metrics", which are used to measure the state of the software or the adequacy of the testing.

### 9.6 Test Cases:

#### Results:

1. **Test Case - Source Image**
   - **Input:** Verify whether image is within the system size
   - **Process:** If size doesn’t match with system size. Then error message is displayed & the output will be invalid.
   - **Output:** If image is of same size display in our application

2. **Test case: Browse-Folder link**
   - **Input:** To Verify the Hyper Links available at left side are working or not
   - **Process:** When we click the link then open folder dialog box should be displayed.
   - **Output:** Set image which is selected from open file dialog

3. **Test case: Browse –File link.**
   - **Input:** To Verify the Hyper Links available at left side are working or not
   - **Process:** When we click the link then open file dialog box should be displayed.
   - **Output:** Set image which is selected from open file dialog
4. **Test case : Establish Database**
   **Input:** Database file must be restored
   **Process:** In this we need to restore the database backup files
   **Output:** All the data from the backup file is restored

5. **Test case : Establishing Database Connection**
   **Input:** Establishing database connection
   **Process:** will give database name, server name and security i.e. If Sql server is under windows authentication will use integrated security as true and sql authentication than will give username and password set for the sql server.
   **Output:** our application gets connection to the database server.

6. **Test case : To add extra images in to our application.**
   **Input :** Will add more images in the database
   **Process :** will add more images in our application for comparisons by using store button or functionality provided in our application. Which in turn connects to database and fills in the images in the database, which the user adds.
   **Output :** the resultant images will be displayed by taking the input image as comparison image.

7. **Test case : Viewing graph for the image comparison output**
   **Input :** will give images that is input image and output image as input for graph output.
   **Process :** we are using graph class for showing the clear comparison between input image and the output image. Here we are going to use graph component class that is graph.cs class.
   **Output :** will get the graph showing comparison between input image and the output image with two different colors  showing the comparisons.
8. **Test case**: Analyzing the output comparison.
   **Input**: our application takes input image and all the possible compared images as input.
   **Process**: will process input image versus all the possible compared images, and finds out what percentage the input image matches with the compared images.
   **Output**: output of input image and the respective compared images will be displayed with percentage match.

9. **Test case**: Hiding the graph.
   **Input**: our application takes graph as input.
   **Process**: makes the graph as invisible from visible mode.
   **Output**: the graph will be hidden from the end user.

10. **Test case**: Exit functionality.
    **Input**: click on exit button will the input for this functionality.
    **Process**: the application will be made to exit by using the application.exit() method in the respective button click.
    **Output**: application will be closed.
10. SYSTEM IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Implementation is the process of converting a new system design into operation. It is the phase that focuses on user training, site preparation and file conversion for installing a candidate system. The important factor that should be considered here is that the conversion should not disrupt the functioning of the organization.

10.1 Scope for Future Enhancement:

The project has covered almost all the requirements. Further requirements and improvements can easily be done since the coding is mainly structured or modular in nature. Improvements can be appended by changing the existing modules or adding new modules.

10.2 Literature Survey:

Human interactive systems have attracted a lot of research interest in recent years, especially for content-based image retrieval systems. Contrary to the early systems, which focused on fully automatic strategies, recent approaches have introduced human-computer interaction. In this paper, we focus on the retrieval of concepts within a large image collection. We assume that a user is looking for a set of images, the query concept, within a database. The aim is to build a fast and efficient strategy to retrieve the query concept.

In content-based image retrieval (CBIR), the search may be initiated using a query as an example. The top rank similar images are then presented to the user. Then,
the interactive process allows the user to refine his request as much as necessary in a relevance feedback loop. Many kinds of interaction between the user and the system have been proposed, but most of the time, user information consists of binary labels indicating whether or not the image belongs to the desired concept. Positive labels indicate relevant images for the current concept, and the negative labels irrelevant images.

To achieve the relevance feedback process, the first strategy focuses on the query concept updating. The aim of this strategy is to refine the query according to the user labeling. A simple approach, called query modification, computes a new query by averaging the feature vectors of relevant images. Another approach, the query reweighting, consists of computing a new similarity function between the query and any picture in the database. A usual heuristic is to weigh the axes of the feature space. In order to perform a better refinement of the similarity function, optimization-based techniques can be used. They are based on a mathematical criterion for computing the reweighting, for instance Bayes error, or average quadratic error. Although these techniques are efficient for target search and monomodal concept retrieval, they hardly track complex image concepts. Performing an estimation of the query concept can be seen as a statistical learning problem, and more precisely as a binary classification task between the relevant and irrelevant classes.

In image retrieval, many techniques based on statistical learning have been proposed, as for instance Bayes classification, k-Nearest Neighbors, Gaussian mixtures, Gaussian random fields, or support vector machines. In order to deal with complex and multimodal concepts, we have adopted a statistical learning approach. Additionally, the possibility to work with kernel functions is decisive.

However, a lot of these learning strategies consider the CBIR process as a classical classification problem, without any adaptations to the characteristics of this context. For instance, some discriminative classifiers exclusively return binary labels when real values are necessary for CBIR ranking purposes. Furthermore, the system has to handle classification with few training data, especially at the beginning of the search, where the query concept has to be estimated in a database of thousands of images with only a few examples. Active learning strategies have been proposed to handle this type of
problem. Another point concerns the class sizes, since the query concept is often a small subset of the database. In contrast to more classical classification problems, relevant and irrelevant classes are highly imbalanced (up to factor 100). Depending on the application context, computational time has also to be carefully considered when an online retrieval algorithm is designed. To address this problem, we assume that any learning task must be at most , where is the size of the database.

Originality of our approach is based on the association of three components:

• boundary correction, which corrects the noisy classification boundary in the first iterations;

• average precision maximization, which selects the images so that classification and mean average precision are enhanced;

• batch selection, which addresses the problem of the selection of multiple images in the same feedback iteration.

We also propose a preselection technique to speed up the selection process, which leads to a computational complexity negligible compared to the size of the database for the whole active learning process. All these components are integrated in our retrieval system, called RETIN.

In this scope, we first present the binary classification and kernel framework to represent complex class distributions. Powerful classification methods and well-defined kernels for global image signatures are evaluated on real database experiments. Secondly, we present active learning to interact with the user, then introduce the RETIN active learning scheme.

Finally, we compare our method to existing ones using real scenario on large databases.
11. CODING

Main.cs

namespace Retrieval
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Diagnostics;
    using System.Drawing;
    using System.IO;
    using System.Windows.Forms;
    using Processing;
    using System.Data.SqlClient;
    using System.Data;
    internal partial class Main : Form
    {
        private bool exit = false;
        public Main()
        {
            this.InitializeComponent();
        }
    }

    #region "GUI delegates"
    private delegate void ProcessImagesDelegate(FileInfo[] files);
    private delegate void SetMaximumDelegate(ProgressBar progressBar, int value);
    private delegate void UpdateOperationStatusDelegate(string format, System.Windows.Forms.Label label, ProgressBar progressBar, int value, DateTime startTime);
    private delegate void UpdateDataGridViewDelegate(BindingList<SimilarityImages> images, DataGridView dataGridView, IList<Button> buttons, TextBox textBox);
    private delegate void DeleteImageDelegate(FileInfo fileInfo);
    #endregion
}
private delegate void ShowGridDelegate(DataGridView dataGridview);
#endregion
#region "Start/stop search"
//THE SEARCH BUTTON
SqlConnection con = new SqlConnection("Database=Img;Data Source=ESTHER-PC;user id=sa;password=coign");
private void FindButton_Click(object sender, EventArgs e)
{
    visible();
    string folder = this.folderTextBox.Text;
    string filee = this.filetext.Text;
    DirectoryInfo directoryInfo;
    FileInfo[] files;
    try
    {
        directoryInfo = new DirectoryInfo(folder
        files = directoryInfo.GetFiles("*.jpg", SearchOption.AllDirectories);
    }
    catch (DirectoryNotFoundException)
    {
        MessageBox.Show("Path not valid.", "Invalid path",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        this.folderTextBox.Enabled = true;
        return;
    }
    catch (ArgumentException)
    {
        MessageBox.Show("Path not valid.", "Invalid path",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
}
#endregion
ProcessImagesDelegate processImagesDelegate = new ProcessImagesDelegate(ProcessImages);
processImagesDelegate.BeginInvoke(files, null, null);
private void CancelButton_Click(object sender, EventArgs e)
{
    this.exit = true;
    this.findButton.Enabled = true;
    this.cancelButton.Enabled = !this.findButton.Enabled;
    this.folderTextBox.Enabled = this.findButton.Enabled;
}
#endregion
#region "GUI delegates body"
private UpdateDataGridViewDelegate updateDataGridViewDelegate =
delegate(BindingList<SimilarityImages> images, DataGridView dataGridView,
IList<Button> buttons, TextBox textBox)
{
    images.RaiseListChangedEvents = true;
    dataGridView.DataSource = images;
    dataGridView.Columns["Similarity"].DisplayIndex = 0;
    dataGridView.Columns["Source"].DisplayIndex = 1;
    dataGridView.Columns["Destination"].DisplayIndex = 2;
    if (images.Count > 0)
        dataGridView.Rows[0].Selected = true;
    buttons[0].Enabled = true;
    buttons[1].Enabled = !buttons[0].Enabled;
    textBox.Enabled = buttons[0].Enabled;
};
SetMaximumDelegate setMaximumDelegate = delegate(ProgressBar
progressBar, int value)
{
progressBar.Maximum = value;

UpdateOperationStatusDelegate updateOperationStatusDelegate =
delegate(string format, System.Windows.Forms.Label label, ProgressBar progressBar, int value, DateTime startTime)
{
progressBar.Value = value;
double percentage = Math.Round(((double)progressBar.Value / (double)progressBar.Maximum), 3);
format += " {0}/{1} ({2}) Elapsed: {3} Estimated: {4}";
TimeSpan elapsed = DateTime.Now.Subtract(startTime);
elapsed = new TimeSpan(elapsed.Days, elapsed.Hours, elapsed.Minutes, elapsed.Seconds, 0);
long estimatedTicks = (elapsed.Ticks / value) * progressBar.Maximum;
TimeSpan estimated = new TimeSpan(estimatedTicks);
estimated = new TimeSpan(estimated.Days, estimated.Hours, estimated.Minutes, estimated.Seconds, 0);
};
ShowGridDelegate showGridDelegate = delegate(DataGridView dataGridView)
{
dataGridView.ResumeLayout();
dataGridView.Enabled = true;
};
#endregion
BindingList<SimilarityImages> similarityImages;
string[] fAllPaths = new string[50];
int findex = 0;
private void ProcessImages(FileInfo[] files)
{


List<ComparableImage> comparableImages;
comparableImages = new List<ComparableImage>();
this.Invoke(setMaximumDelegate, new object[]
{
    this.workingProgressBar, files.Length
});
int index;
index = 0;
DateTime operationStartTime;
//my code
FileInfo f = new FileInfo(openFileDialog1.FileName);
ComparableImage comparableImage1 = new ComparableImage(f);
comparableImages.Add(comparableImage1);
//end my code
operationStartTime = DateTime.Now;
foreach (FileInfo file in files)
{
    if (this.exit)
    {
        return;
    }
    //comparableImages comparebleimaaa = new ComparableImage(file);
    this.Invoke(updateOperationStatusDelegate, new object[]
    { "Processed images", workingLabel, workingProgressBar, index,
        operationStartTime });
}
//my code to retrieve from database
SqlDataAdapter sda;
DataSet ds = new DataSet();
Byte[] newbyte = new Byte[0];
MemoryStream m;
int k = 0;
sda = new SqlDataAdapter("select * from imagetab", con);
sda.Fill(ds);
Image x;
FileInfo fnew;
string fpath;
foreach (DataRow dr in ds.Tables[0].Rows)
{
    newbyte = (Byte[])dr[1];
    m = new MemoryStream(newbyte);
    x = Image.FromStream(m);
    x.Save("c:\windows\" + dr[2].ToString());
    fpath = "c:\windows\" + dr[2].ToString();
    //comparableImages comparableimageaaa = new ComparableImage(file);
    //
}
pictureBox1.Image = Image.FromStream(m);
//end mycode to retrieve from database
this.Invoke(this.setMaximumDelegate, new object[]
{
    this.workingProgressbar, (comparableImages.Count
    +(comparableImages.Count -1)) / 2
});
index = 0;
List<SimilarityImages> similarityImagesSorted = new List<SimilarityImages>();
operationStartTime = DateTime.Now;
string[] selectedFiles = openFileDialog1.FileNames;
double similarity = source.CalculateSimilarity(destination);
destination, similarity);
object[]
"Compared images", workingLabel, workingProgressBar,
index, operationStartTime }
);

similarityImages = new BindingList<SimilarityImages>(similarityImagesSorted);
List<Button> buttons2 = new List<Button>();
bubble2.Add(findButton);
bubble2.Add(cancelButton);
this.BeginInvoke(updateDataGridViewDelegate, new object[] { similarityImages, imagesDataGridView, buttons2, folderTextBox });
}
private void ShowProjections(Graph graph, ComparableImage comparable)
{
    graph.ClearCurves();
    graph.AddPlotCurve(Color.Red, comparable.Projections.HorizontalProjection);
    graph.AddPlotCurve(Color.Green, comparable.Projections.VerticalProjection);
    graph.Invalidate();
}
protected override void OnClosing(EventArgs e)
{
    exit = true;
}
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    this.folderBrowserDialog.ShowDialog();
    this.folderTextBox.Text = this.folderBrowserDialog.SelectedPath;
}
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
}
private void ShowSelectedImages()
if (imagesDataGridView.SelectedRows.Count <= 0)
{
    return;
}

SimilarityImages sim = (SimilarityImages)
imagesDataGridView.SelectedRows[0].DataBoundItem;
FileStream streamDestination = new
System.IO.FileStream(sim.Destination.File.FullName, FileMode.Open,
FileAccess.Read);
Image source = Image.FromFile(openFileDialog1.FileName);
{3}"
Image destination = Image.FromStream(streamDestination);
streamDestination.Close();
string infoFormat = "Resolution: {0}x{1}\nSize: {2} kb\nFull path:
string resolutionFormat = "{0} ({1}x{2})";
sourcePictureBox.Image = Image.FromFile(openFileDialog1.FileName);
sourceLabel.Text = string.Format(resolutionFormat, "Source",
source.Width, source.Height);
destinationPictureBox.Image = destination;
mainToolTip.SetToolTip(destinationPictureBox, string.Format(infoFormat,
destination.Width, destination.Height, Math.Round((double)
destinationLabel.Text = string.Format(resolutionFormat,
"Destination", destination.Width, destination.Height);
ShowProjections(destinationGraph, sim.Destination
ShowProjections(sourceGraph, sim.Source);
}
private void DeleteSourceLinkLabel_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
if (imagesDataGridView.SelectedRows.Count <= 0)
{
    return;
}

DialogResult result = MessageBox.Show("Delete the image?", "Confirm delete", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
if (result == DialogResult.Cancel)
{
    return;
}

SimilarityImages sim = (SimilarityImages)imagesDataGridView.SelectedRows[0].DataBoundItem;
DeleteImageDelegate deleteImageDelegate = new DeleteImageDelegate(DeleteImage);
imagesDataGridView.SuspendLayout();
imagesDataGridView.Enabled = false;
deleteImageDelegate.BeginInvoke(sim.Source.File, new AsyncCallback(ShowDataGrid), null);
}

private void ShowDataGrid(IAsyncResult result)
{
    this.Invoke(showGridDelegate, imagesDataGridView);
}

private void DeleteDestinationLinkLabel_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    if (imagesDataGridView.SelectedRows.Count <= 0)
        return;

    DialogResult result = MessageBox.Show("Delete the image?", "Confirm delete", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
    if (result == DialogResult.Cancel)
return;
SimilarityImages sim = (SimilarityImages)imagesDataGridView.SelectedRows[0].DataBoundItem;
DeleteImageDelegate deleteImageDelegate = new
DeleteImageDelegate(DeleteImage);
imagesDataGridView.SuspendLayout();
imagesDataGridView.Enabled = false;
deleteImageDelegate.BeginInvoke(sim.Destination.File, new
AsyncCallback(ShowDataGrid), null);
}  
private void DeleteImage(FileInfo fileInfo)
{
    try
    {
        List<SimilarityImages> toDelete = new List<SimilarityImages>();
        for (int index = 0; index < similarityImages.Count; index++)
        {
            SimilarityImages item = similarityImages[index];
            if ((item.Source.File.FullName.Equals(fileInfo.FullName,
                StringComparison.InvariantCultureIgnoreCase)) ||
                (item.Destination.File.FullName.Equals(fileInfo.FullName,
                StringComparison.InvariantCultureIgnoreCase)))
                toDelete.Add(item);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
private void OpenSourceLinkLabel_LinkClicked(object sender,
if (imagesDataGridView.SelectedRows.Count <= 0)
return;
SimilarityImages sim = (SimilarityImages)
this.imagesDataGridView.SelectedRows[0].DataBoundItem;
Process.Start(sim.Source.File.FullName);
}
private void OpenDestinationLinkLabel_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
if (this.imagesDataGridView.SelectedRows.Count <= 0)
{
return;
}
SimilarityImages sim = (SimilarityImages)
this.imagesDataGridView.SelectedRows[0].DataBoundItem;
Process.Start(sim.Destination.File.FullName);
}
private void DataGridView1_SelectionChanged(object sender, EventArgs e)
{
}
this.ShowSelectedImages();
private void selectFolderToSerachToolStripMenuItem_Click(object sender, EventArgs e)
{
gbxFolder.Visible = true;
}
public void visible()
{
workingLabel.Visible = true;
}
private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
{
    about ab = new about();
    ab.Show();
}

private void closeToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void viewGraphToolStripMenuItem_Click(object sender, EventArgs e)
{
    label2.Visible = true;
    label3.Visible = true;
    sourceGraph.Visible = true;
    destinationGraph.Visible = true;
}

private void hideGraphToolStripMenuItem_Click(object sender, EventArgs e)
{
    label2.Visible = false;
    label3.Visible = false;
    sourceGraph.Visible = false;
    destinationGraph.Visible = false;
}

public string ss;
private void linkLabel1_LinkClicked_1(object sender,
LinkLabelLinkClickedEventArgs e)
{
if (openFileDialog1.ShowDialog() == DialogResult.OK)
{
this.filetext.Text = openFileDialog1.FileName;
sourcePictureBox.Image = Image.FromFile(openFileDialog1.FileName);
}
}
private void Main_Load(object sender, EventArgs e)
{
}
private void sourceGraph_Click(object sender, EventArgs e)
{
}
public int index1;
private void openFileDialog1_FileOk(object sender, CancelEventArgs e)
{
DirectoryInfo directoryInfo = new DirectoryInfo(folderTextBox.Text);
FileInfo[] files = directoryInfo.GetFiles("*.jpg",
SearchOption.AllDirectories);
string searchImage = openFileDialog1.FileName.ToString();
int se = searchImage.LastIndexOf("\";
int aaa = searchImage.Length - se;
string aa = searchImage.Substring(se + 1, aaa - 1);
int count = files.Length;
index1 = 0;
for (int i = 0; i <= count-1; i++)
{
}
}
private void folderBrowserDialog_HelpRequest(object sender, EventArgs e) {
}

private void gbxFolder_Enter(object sender, EventArgs e) {
}

private void mainToolTip_Popup(object sender, PopupEventArgs e) {
}

SqlCommand cmd = new SqlCommand();
Byte b2;
DirectoryInfo directoryInfo;
FileInfo[] filesnew;
Image i;

private void button1_Click(object sender, EventArgs e) {
    FolderBrowserDialog fb = new FolderBrowserDialog();
    fb.ShowDialog();
directoryInfo = new DirectoryInfo(fb.SelectedPath);
filesnew = directoryInfo.GetFiles("*.jpg", SearchOption.AllDirectories);
foreach (FileInfo file in filesnew) {
    cmd.CommandText = "insert into imagetab values(@img field, @imagename)";
FileStream f = new FileStream(file.FullName, FileMode.Open, FileAccess.Read);
Byte[] b = new Byte[f.Length];
f.Read(b, 0, Convert.ToInt32(f.Length));
f.Close();
SqlParameter p = new SqlParameter("@imgfield", SqlDbType.VarBinary, b.Length, ParameterDirection.Input, false, 0, 0, null, DataRowVersion.Current, b);
SqlParameter p1 = new SqlParameter("@imagename", SqlDbType.VarChar, 50);
p1.Value = file.Name;
}
MessageBox.Show("Images Stored Successfully");
}
private void Main_FormClosing(object sender, FormClosingEventArgs e)
{
FileInfo fi;
for (int z = 0; z < fAllPaths.Length; z++)
{
try
{
if (fAllPaths[z] != null)
{
fi = new FileInfo(fAllPaths[z]);
fi.Delete();
}
}
catch(Exception ex)
{
}
}
12. OUTPUT SCREENS

12.1 Main Screen

![Image Retrieval Screen](image)

Fig: 12.1

**Screen Description:**

This is the main screen of the project Active Learning Methods For Interactive Retrieval. The menu bar has four menu objects File, Help, Graph and Window.
12.2 Selecting About Option

*Fig: 12.2*

**Screen Description:**

The Help menu object has a menu item named about and the selection of that menu item is displayed in this screen shot.
12.3 About Screen

Screen Description:

After selecting the About menu item, the About Screen will be displayed.
12.4 Selecting Search for Folder Option

Screen Description:

The File menu object has a menu item named Select folder to search and the selection of that menu item is displayed in this screen shot.
12.5 Selecting Source Image

Fig: 12.5

Screen Description:

On selecting the browse link beside the select file option a sub window which contains all the files will be displayed and we have to select our source image from that sub window.
12.6 After Selecting Training Set and Source Image

Fig: 12.6

Screen Description:

The above screen shot displays the paths of the Training set and the Source Image.
12.7 On Clicking The Search Button

**Screen Description:**

On clicking the Search button all the computations take place and the result is displayed in the form of a grid.
12.8 Selecting View Graph Option

Screen Description:

The Graph menu object has two menu items named View graph and Hide graph. The selection of the View Graph is shown in this screen shot.
Screen Description:

On selecting the View graph menu item the source image and destination image graphs are displayed.
12.10 Selecting Hide Graph Option

Screen Description:

The Graph menu object has two menu items named View graph and Hide graph. The selection of the Hide Graph is shown in this screen shot.

Fig: 12.10

<table>
<thead>
<tr>
<th>Source</th>
<th>Quality</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.jpg</td>
<td>Avg. 10pp</td>
<td>a.jpg</td>
</tr>
<tr>
<td>a.jpg</td>
<td>Avg. 10pp</td>
<td>a.jpg</td>
</tr>
<tr>
<td>a.jpg</td>
<td>10pp</td>
<td>a.jpg</td>
</tr>
<tr>
<td>a.jpg</td>
<td>10pp</td>
<td>a.jpg</td>
</tr>
<tr>
<td>a.jpg</td>
<td>10pp</td>
<td>a.jpg</td>
</tr>
<tr>
<td>a.jpg</td>
<td>10pp</td>
<td>a.jpg</td>
</tr>
<tr>
<td>a.jpg</td>
<td>10pp</td>
<td>a.jpg</td>
</tr>
</tbody>
</table>

73
12.11 On Selecting Hide Graph Option

Screen Description:

On selecting the Hide graph menu item the source image and destination image graphs will not be displayed.
12.12 Selecting Option To Exit From The Window

Fig: 12.12

Screen Description:

On selecting the exit option we can close the window of the project.
13. CONCLUSION

The RETIN active learning strategy for interactive learning in content-based image retrieval context is presented. The classification framework for CBIR is studied and powerful classification techniques for information retrieval context are selected. After analyzing the limitation of active learning strategies to the CBIR context, we introduce the general RETIN active learning scheme, and the different components to deal with this particular context.

The main contributions concern the boundary correction to make the retrieval process more robust, and secondly, the introduction of a new criterion for image selection that better represents the CBIR objective of database ranking. Other improvements, as batch processing and speed-up process are proposed and discussed. Our strategy leads to a fast and efficient active learning scheme to online retrieve query concepts from a database. Experiments on large databases show that the RETIN method gives very good results in comparison to several other active strategies.

The framework introduced in this article may be extended. We are currently working on kernel functions for object classes retrieval, based on bags of features: each image is no more represented by a single global vector, but by a set of vectors. The implementation of such a kernel function is fully compatible with the RETIN active learning scheme described in this article, and the initial results are really encouraging.

**Future Scope of the Project:**

The project has covered almost all the requirements. Further requirements and improvements can easily be done since the coding is mainly structured or modular in nature. Improvements can be appended by changing the existing modules or adding new modules.
BIBLIOGRAPHY

1. R. Veltkamp, Content-based image retrieval system: A survey

2. N. Najjar, J. Cocquerez, and C. Ambroise, Feature selection for semi supervised learning applied to image retrieval

3. Date, C. J, An Introduction to Database Systems

4. S. Tong and E. Chang, Support vector machine active learning for image retrieval

5. N. Doulamis and A. Doulamis, Image Processing

6. Roger S. Pressman, Software Engineering, A practitioner’s Approach

7. Donald Thompson and Rob S. Miles, Programming with the Microsoft .NET Micro Framework


