# Chapter 1

**Introduction**

Content Based Image Retrieval has been researched for many years now as an alternative of text based image retrieval. Increase in communication bandwidth, information content and the size of the multimedia databases has given rise to the concept of content based image retrieval. **Content-based image retrieval** (**CBIR**), also known as **query by image content** (**QBIC**) and **content-based visual information retrieval** (**CBVIR**) is the application of computer vision techniques to the image retrieval problem, that is, the problem of searching for digital images in large databases. "Content-based" means that the search analyzes the contents of the image rather than the metadata such as keywords, tags, or descriptions associated with the image. The term "content" in this context might refer to colors, shapes, textures, or any other information that can be derived from the image itself. CBIR is desirable because searches that rely purely on metadata are dependent on annotation quality and completeness. Having humans manually annotate images by entering keywords or metadata in a large database can be time consuming and may not capture the keywords desired to describe the image. The evaluation of the effectiveness of keyword image search is subjective and has not been well-defined. In the same regard, CBIR systems have similar challenges in defining success

Current research works attempt to obtain and use the semantics of image to perform better retrieval. There exist many systems for image retrieval but efficient CBIR is a challenging task. Some works focused on how to represent an image, which means how to extract the visual features of an image. Some other works focused on how to understand an image, which means how to extract the objects in an image and describe the relationship between objects. All these works emphasize on the accuracy of the retrieval, but pay very little attention on the ability of responding to a huge amount of requests. However the content based image retrieval is very time consuming due to the extraction and matching of high dimensional and complex features. The aim of this project is to achieve highly accurate results along with a reduced computation time.

### Existing System Analysis

Initially, image retrievals used the content from an image individually. Approaches using a combination of contents then started gaining prominence a bit later. Combining shape and color using various strategies such as weighting, histogram-based, kernel based, or invariance-based has been one of the premier combination strategies. Shape and texture using elastic energy based approach to measure image similarity has been presented .Others include an automated extraction of color and texture information using binary set representations, a color histogram along with the texture and spatial information. Image retrieval by segmenting them had been the focus of few research papers. Despite the extensive research effort, the retrieval techniques used in CBIR systems lag behind the corresponding techniques in today’s best text search engines, such as in query, Alta Vista, Lycos, etc. For CBIR the research is primarily focused on exploring various feature representations, hoping to find a “best” representation for each feature.

### Aims and Objective

Search Engines are always evolving and an area of great research scope. However all of the image retrieval techniques we studied had some drawbacks and each and every paper aims at reducing those drawbacks. The aim of this project is to try and improvise on two of the most recent search techniques and strive to attain a high level of efficiency.

### Organization of the Report

The chapter 1 contains the problem statement which stated the goal of entire project. This includes the aim and objectives for the same. Chapter 2 consists of the literature survey tells us about the existing system and also how the idea of question paper generation is implemented nowadays. The chapter 3 contains the details of nature and scope of the proposed work in terms of its objectives and methodologies used along with issues, hardware and software requirements. Later, in chapter 3 the entire system design is given. The chapter 4 gives the implementation of the system which helps to understand the system more clearly. In chapter 5 will discuss about the

future scope and finally chapter 6 concludes report.

**Chapter 2**

**Literature Survey**

In the following literature survey we gave gathered various research on the CBIR systems being used. These papers explain various advantages and disadvantages of the methods and give a gist of the topic.

**2.1 Research Papers Survey**

**Paper[1].** Jun Yue, Zhebo Li, Lu Liu, Zetian Fu, discussed about global color histogram and block color histogram. Histograms for HSV are generated for both these and the results are compared. Co-occurrence matrix and texture features (gray scale value) is used to find grey scale quantification which is used for image retrieval. Both these techniques are fused together to build the final CBIR.

**Paper[2].** A. Ramesh Kumar, D. Saravananan, discussed a two-stage CBIR approach to reduce the computation effort and make the retrieval fast is proposed. The image is first converted into the desired color space. Then the feature vector derived from the color histograms of the image is used to generate a grid code via vector quantization, so as to group the database images into grids, each of which contains the same grid code and visually similar patterns.

**Paper[3].** Bryan Anenberg, Michael Meister, discussed how the goal of foreground-background segmentation is to divide the pixels of an image into exactly two sections, one foreground and one background. Foreground-background segmentation is a challenging task, and has many applications in object recognition and classification

**Paper[4].** Thomas Malmer, discussed an iterative algorithm that combines statistics and Graph Cut in order to accomplish detailed 2D segmentation with very limited input, originally developed at Microsoft Research Cambridge, UK. The work flow for the user goes as follows: The user selects a image which should be used to perform the segmentation on and draws a rectangle around the object of interest that should be segmented. That's all input that is needed by the GrabCut algorithm to perform a segmentation. Afterwards the user can do a touch up of the image if needed or use the image again input by selecting a new rectangle.

**Paper[5].** Sheikh Aamir Rasool, Amarinder Kaur, discussed extracting color mean features and also color standard deviation feature. The method used consists of HMMD (Hue Min Max Difference) color plane. It is proved that HMMD along with color mean features and color standard deviation feature tends to reduce the size of feature vectors, storage space and gives high performance than, RGB color mean feature. Further, HMMD color space model is used to improve the feature extraction and improve the precision.

**Paper[5].** Harkirat Kaur, Baljit Singh Khehra, discussed how HMMD (Hue Min Max Difference) color space, is used in order to efficiently describe the colors in an image. The hue has the same and minimum among the R, G, and B values, respectively. The diff component is defined as the difference between max and min. Only three of the four components are sufficient to describe the HMMD space. It is observed that the HMMD color space is very effective and compared favorably with the HSV color space.

# Paper[6]. Pradnya Vikhar, Pravin Karde, discussed how the basic CBIR system is developed by combining features like color moments, color-correlogram and Gabor texture features along with edge histogram descriptor. Further the results obtained are compared with CBIR system using SVM classifier.

**2.2 Analysis**

These papers indicate that there is a myriad of techniques to perform image retrieval. However, these techniques have some drawbacks and image retrieval isn’t as sophisticated as text retrieval. Some techniques have a lot of issues with precision wherein the expected result is not attained. The techniques which have a high precision have a very large computation time. Thus we need to strive to develop a system which takes into account both these factors without making any compromises. The detailed proposal for the system is mentioned in the next chapter

# Chapter 3

# Proposed System

In this chapter we will discuss about the design of the system, system requirements, modules present in the system, implementation methodology and all the planning that has to be done before beginning the implementation of the project.

### Problem Statement:

The Edge histogram method has low precision in certain cases when the segmentation cannot be done properly. Hence, this system will move Edge histogram descriptor to a second level filter and thus allowing the first level filter to pass on a refined dataset for the EHD thus tackling the precision issue. The HMMD (Hue Min Max Difference) is preferred over other techniques as it converts a problem based on the well-known RGB histogram to the HMMD histogram table thus reducing computation time. Also this technique concentrates on the images with 32 and 128 bins only and thus filtering out the need for unnecessary scaling and conversion. For the EHD the image is scaled down by a factor of two, thus reducing computation time and not affecting the image search. These two methods have never been used in unison before and we intend to try and use them to produce efficiency in output. However, merging two completely different approaches is always a difficult task as while both are working together, the results of one should not hamper the results of other in an unwanted way.

### Proposal

Edge Histogram Descriptor is used to perform two dimensional segmentation in an image. The selection of the image is done by generating a matrix to bring out the area of interest. This helps to get rid of unwanted objects and to make the search more precise.Many methods such as magic wand, intelligent scissors, knockout and graphcut are also available but the reason for choosing EHD is due to its precision and is user friendly. This technique is only recently being used for CBIR.

String based color comparison method for content based image retrieval is based on support vector machine classifier. In this method the feature extraction is done based on the color string coding followed by string comparison. This technique transfers image retrieval problem to strings comparison.

### Proposed Implementation

* + - * 1. First, user builds the database using calcfeatures which gives us different tables in the features matlab file.
        2. The image is scaled down by a factor of two and edge value is computed.
        3. The RGB feature histogram is generated for the image.
        4. The RGB to HMMD model uses RGB values and returns Hue, Sum and Difference.
        5. These values are then converted into a quant table of hue, sum and difference.
        6. If the image does not have a histogram of 32 or 128 bins an error message is displayed on the command line.
        7. After calculating the HMMD values the histogram is created.
        8. When the query image is input, it calls the find similar function.
        9. The database is loaded in the background to avoid processing delays.
        10. The distance between all the image features in terms of HMMD and EHD are calculated.
        11. The output is generated based on the ascending order of these distances.

### Design of System

### Overall System

Image database

Input query Image

Processing of image

Extraction of required image features

Edge Histogram Descriptor

Feature Database

HMMD Color Space Model

Query image features

Image matching by comparing distances between features

Final Results

Fig.3.1.2.1. Overall System

The image is given as a query to the system. After selecting the query user gets the option to perform search. Preprocessing takes places and all the required information is extracted from the image. These extracted features are then compared to the database containing features of the image dataset. The output is obtained on the same GUI window with images shown in descending order of proximity.

### Hue Min Max Difference (HMMD)

### Hue Min Max Difference color model is a patented color quantization technique. This techniques is based upon factors like hue, shade, tone, tint, brightness of the color. The colors adjacent to a particular color are considered to be neighboring colors. In the color space the indices are considered as partial values according to the distance. This method was developed to enhance the performance of image searches based on content.

The steps involved are as follows:

Step 1: Image is obtained from training dataset.

Step 2: Calculate the Hue value from image

Step 3: Calculate the Min and Max value from image

Step 4: Calculate the HMMD transform for image

Step 5: Features of image is extracted and save it in training file.

Step 6: If this image is the last image, then preprocess the training file and train the classifier, otherwise go to step 1.

Step 7: Now, image is obtained from testing data set. Step 8: Extract the feature of image. Step 9: If it is the last image then predict the class using trained classifier, otherwise go to step 4.

Input query

Extract from the RGB value of the next pixel

Transformation into HMMD color space model

Quantify color

Global color histogram = global color histogram[color index] + 1

Local color histogram = local color histogram[color index] + 1

last pixel?

No

Yes

Extracted features

Output to dataset

Fig.3.1.2.2. Working of HMMD

### Edge Descriptor

The need to use this technique stems from the lack of spatial information used in the color based techniques. Edge information is used in this method to retrieve images.

Edge Histogram Descriptor (EHD) belongs to the family MPEG-7 descriptors. MPEG-7 provides different standard Multimedia Descriptors to describe and interpret visual contents for images. It offers the interoperability across different multimedia databases. Main visual features descriptors are color, shape and texture descriptors. In proposed methodology, Edge Histogram Descriptor (EHD) is applied to extract texture features of images. The EHD characterizes edges to represent spatial distribution in an image.

Feature extraction process using EHD consists of following steps:

1) First, the array of digital image is spitted into equal 4X4 subparts/ sub-images.

2) In next step, every subpart is further divided into no overlapping square blocks. Here size of blocks are depends on the resolution of input image.

3) From every block the edge is calculated and then type is identified using the filter coefficients illustrated in figure 3.1.2.3.a. There are six types of edges. The type of edge may be vertical, horizontal, 450 diagonal, 1350 diagonal, nodirection edge and no-edge. Initially first five types of edges are identified and no-edge blocks can be automatically obtained after the process of normalization.

4) By applying above steps, from every sub-image 5-bin edge histogram is obtained. Five bins are vertical, horizontal, 450 diagonal, 1350 diagonal, and no-direction

5) Then, the value of each bin in the sub-image is computed by normalizing total number of image blocks in the sub-image.

6) Finally for these normalized bins nonlinear quantization value is calculated to limit the number of bits sufficient for the descriptor. Texture features of input image and images present in database are extracted using above steps.

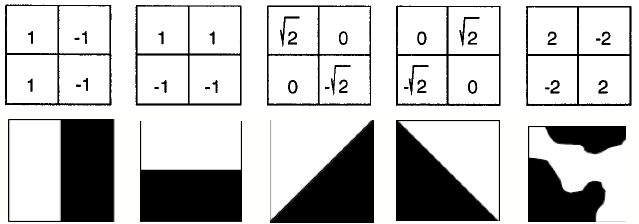


Fig.3.1.2.3.a.Types of Edges

Input Query

Scale/ adjust image as per requirement

Separate non-overlapping square blocks

Calculate edge and obtain histogram

5- bins done?

No

Yes

Output to database

Features extracted

Fig.1.2.3.b.Working of EHD

### Build database code flow

The builddatabase starts retrieving features from the images in a given folder. Depending on the size of the image the processing time per image varies. We use a 2000 image database which is a good number considering other search engines available. This database takes roughly 1.9 seconds \* 2000 = 3800 seconds. i.e roughly an hour to get created.

Features.mat

-csd128hist

-edges

-names

Colordescriptor.m

Hmmdquanttable128.m

Hmmdquanttable32.m

Rgb2hmmd.m

Rgb2quanthmmd.m

ehd.m

Calcfeatures.m

Builddatabase.m

DB

Fig.3.1.3.Build Database code flow

### GUI codeflow

The GUI allows users to select a query image and then runs the image retrieval code on the same. The features of the query image are compared to the features stored in features.mat file and thus we get a result of about 20 similar images as the output. The precision and negative outputs are discussed in the result and analysis section.

Output

Domcolordist.m

Ehdist.m

Features.mat

Calcfeatures.m

Findsimilar.m

SimpleGUI.m

Fig.3.1.4. GUI code flow

### 3.3 System Requirements

In this sub-topic we will be discussing about the software applications, programming languages, system configuration that was needed to develop this project.

### Software Requirements

**Matlab**

The entire project was developed on the MATLAB R2015a edition. All the versions developed post 2013 are efficiently able to run this code. Matlab can be installed on any native OS and the code can be executed. Matlab was chosen over languages like Java, C++ and Python. Matlab is really efficient in dealing with mathematical formulas and there are vast number of algorithms available. Merging all of the algorithms also becomes easy because of its simple syntax. Creating a GUI is also easy with matlab.

### 3.3.2 Hardware Requirements

Hardware requirements of the system are:

* RAM – Minimum 1GB, but 4GB recommended for no lag
* Operation System – Microsoft Windows XP &above
* Processor Speed – Minimum 1GHz
* Hard disk – Minimum 50GB

### Non-functional Requirements

* Efficiency Requirement: The processing should be quick when extracting features. The system should not lag or freeze during execution as image processing takes a lot of CPU overhead.
* Reliability Requirement: The results should be accurate as much as possible. It is impossible to get 100% efficiency but we should strive to reduce the number of negative results.
* Usability Requirement: The UI is kept as simple as possible for even a layman to use. Also the results should be legible enough to see clearly.

**Chapter 4**

**Result and Analysis**

In this chapter, we are going to discuss about the results of the implementation of the proposed system. Modules of this system are implemented in a parallel manner and later on all the modules were merged to produce the final results. In the later part, we will analyze the system by comparing the results by using same database in another feature engines.

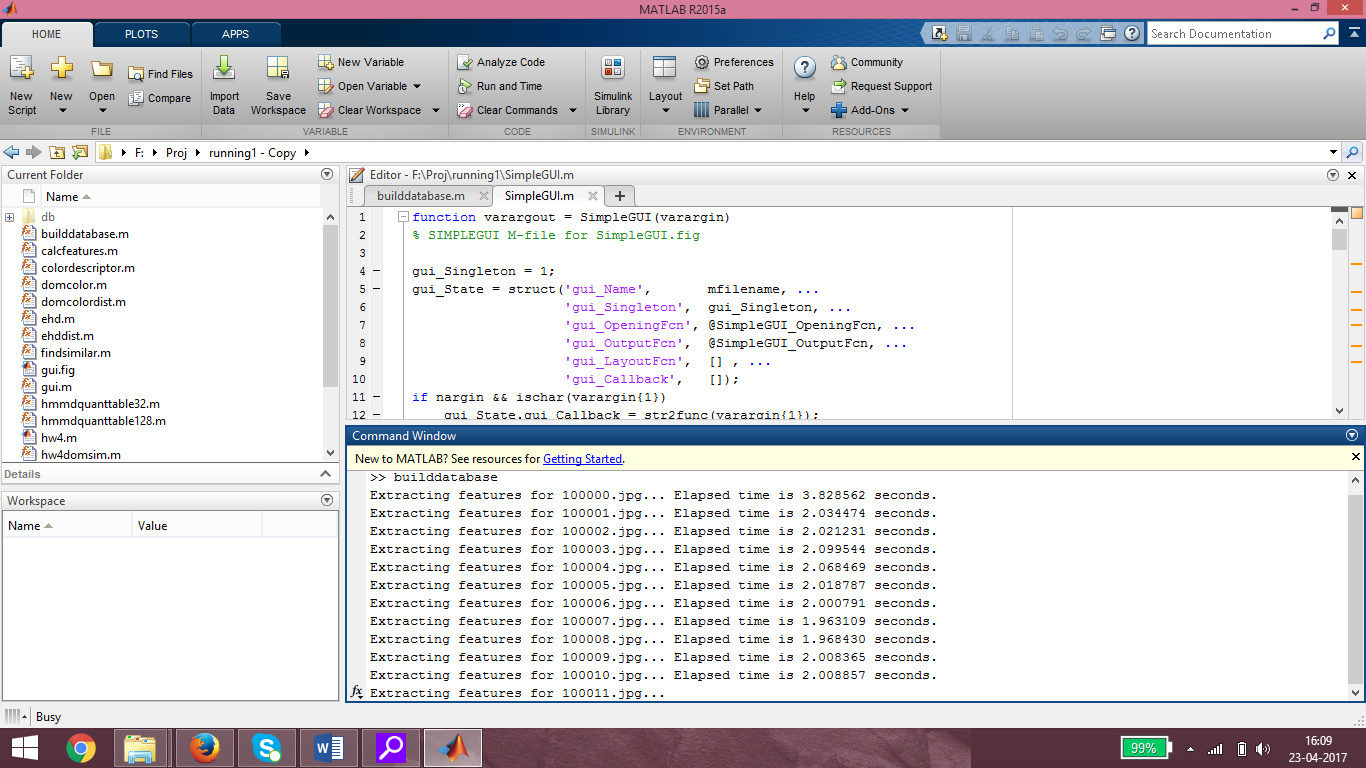
#### Results and Implementation

In this section, output of the implementation of each module will be shown with the help of screenshots in detailed manner which is as follows.

This will enable understanding how effectively it works as well as in what cases do we need better features to improve upon what we build.

#### Build Database

We need to first generate the features.mat database of all the images in the dataset. This is done by running the builddatabase code on the command line. This will run all the modules under calfeatures and the processing time is displayed on the command line.



.

Fig.4.1.1.Build database

##### GUI execution

To run the User interface we have to run it from the command line which will then shoot up the interface window. This is done as follows:

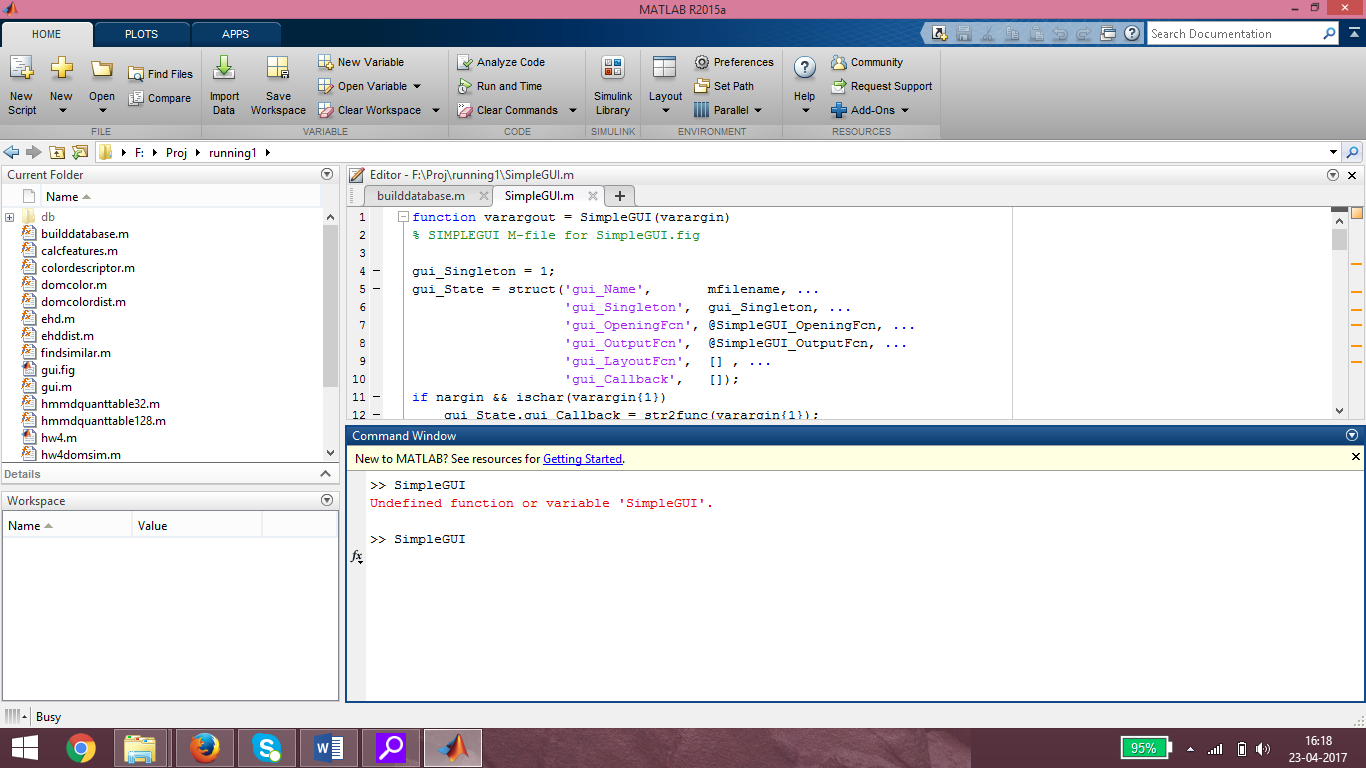


Fig.4.1.2.GUI execution

#### GUI Window

#### The GUI window is kept simple in order for users to understand how to use it without having any background knowledge. This window contains the following options:

#### Close: To close the GUI window.

#### Minimize: To minimize the GUI window.

#### Browse: To select the image which will be used as a query.

#### Go! : To execute the image search.

#### 

Fig.4.1.3. GUI window

#### Browse:

The Browse commands opens up a window which allows users to search for the query image and select it. This window is similar to the selection window which we see while using general windows applications. Hence, it becomes easy to navigate through it.

Users have the options to navigate through directories, select an image, and select the type of images to search for and to cancel the selection.

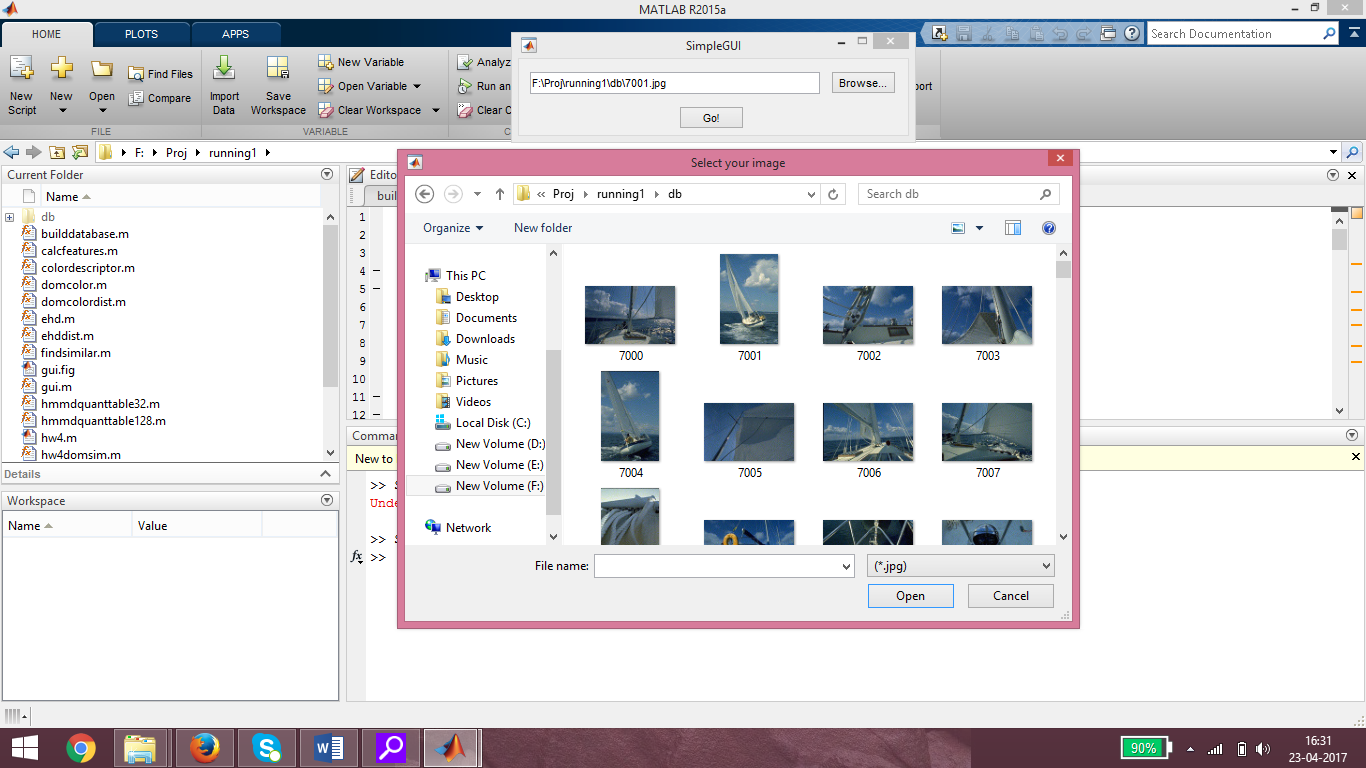


Fig.4.1.4.Browse

**4.1.5 Go!**

The Go! Command runs the image retrieval engine and finally presents the desired output. We have chosen to show 20 images in our output which are followed by the query image. You will observe that if the query is a part of the database it will be shown first in the output because it has the strongest similarity in terms of the features.

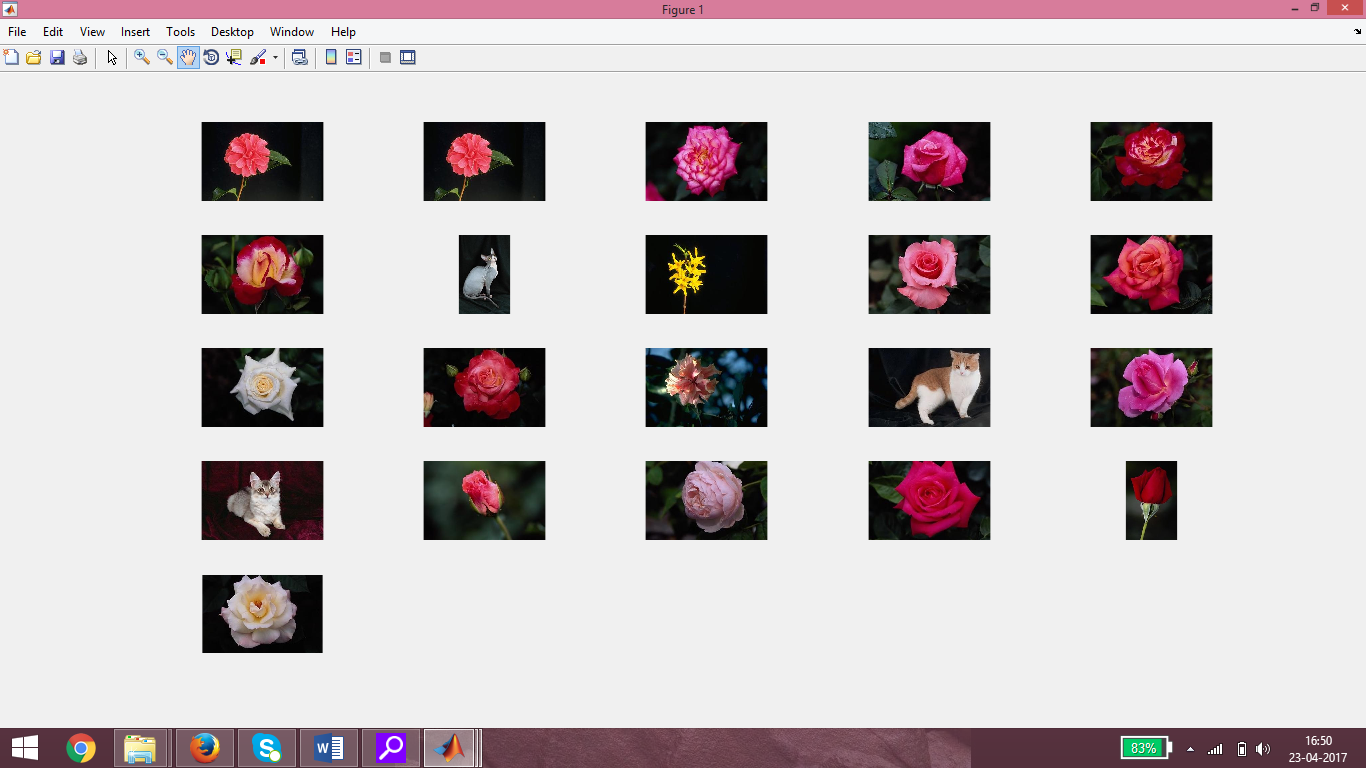
****

Fig.4.1.5. Go!

**4.2 Analysis**

**4.2.1 Preprocessing time**

As observed from the builddatabase figure. The Average time taken for each image can be considered to be 2 seconds.

Our database consists of 2000 images.

Therefore,

Total time to preprocess

= average time per image \* no. of images

= 2 seconds \* 2000

= 4000 seconds

= 66.67 minutes

~ 1 hour 6 minutes.

If the images in the database are of low resolution, it takes lesser processing times. But this dataset gives us a good processing time with efficient images.

**4.2.2 Efficiency analysis**

Positive output: This is the number of images which are expected and are in some way similar to the query image.

Negative output: This is the number of images which are a part of the output but they are in some way not similar to the query image.

On observing our output for multiple queries it is found that if there is too much of some particular color in the query image, the output will give unrelated images with a large amount of the same color.

We analyze how this system fares in comparison to other features which we could have used by using the same database on an open source project. We will compare the number of positive and negative outputs of both.

**Our system (HMMD and EHD)**

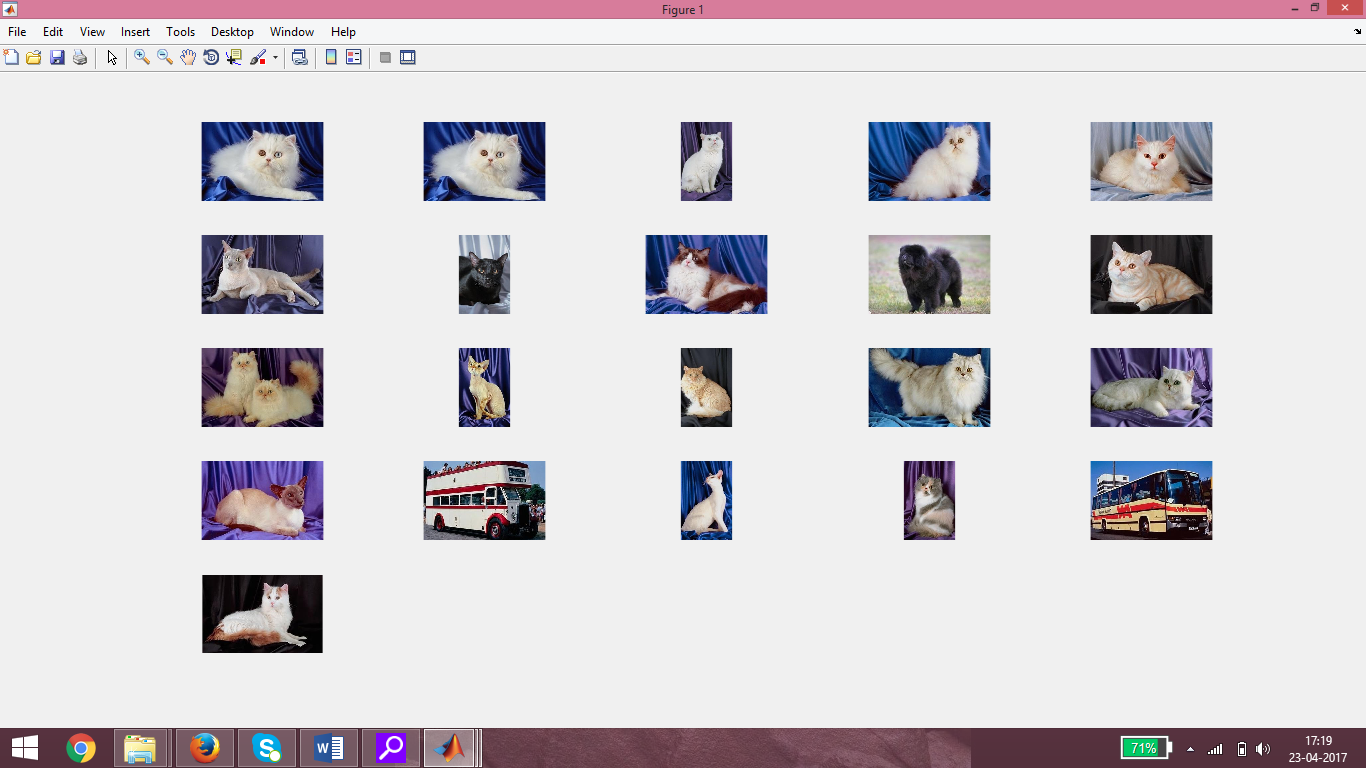


Fig.4.2.2.a.HMMD and EHD

Here we select a random image from the database and use it as a query for the engine. The returned output is then analyzed and the number of positive and negative searches are counted.

Positive images: 17/20

Negative images: 3/20

**RGB and EHD**

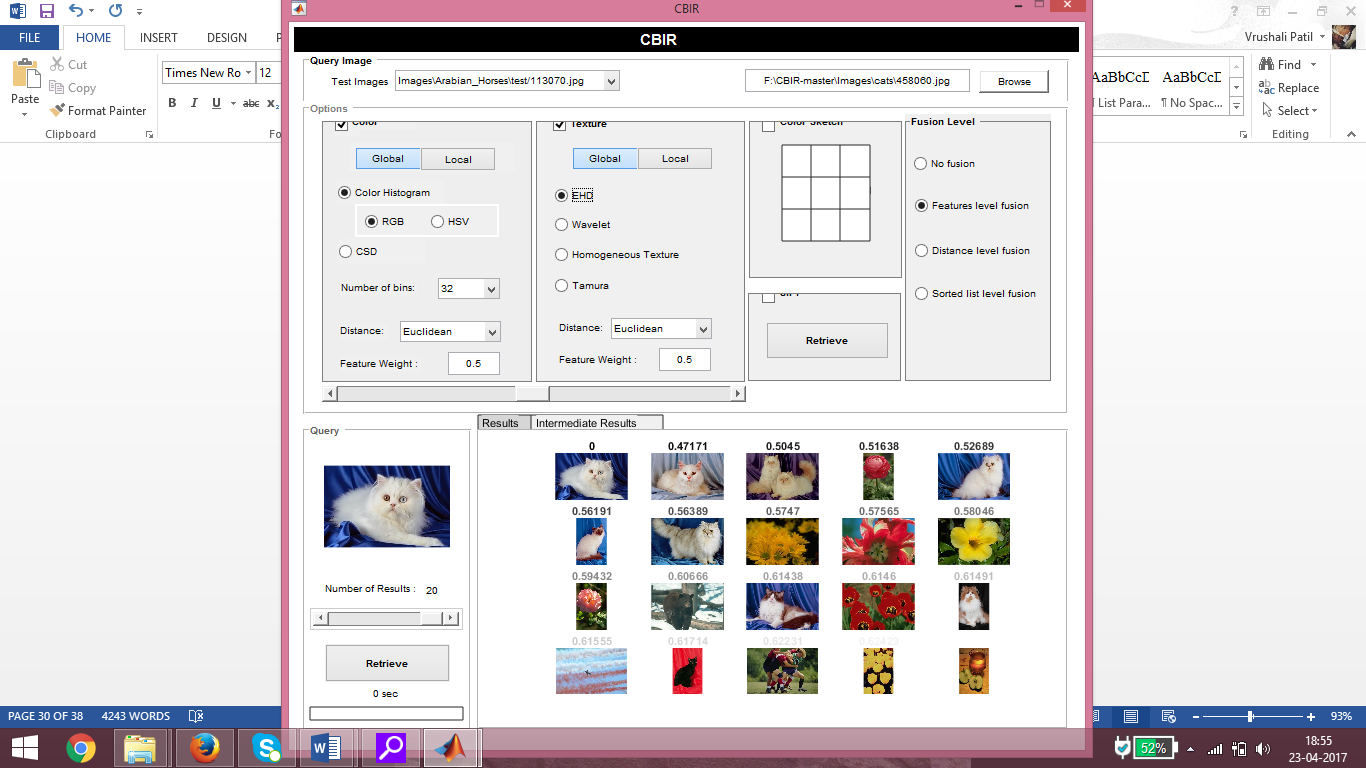


Fig.4.2.2.b.RGB and EHD

We use another CBIR system available by selecting RGB and EHD as the extraction and comparison features. The output is then analyzed and positive and negative images are counted.

Positive Images: 9/20

Negative Images: 11/20

**RGB and Wavelet**

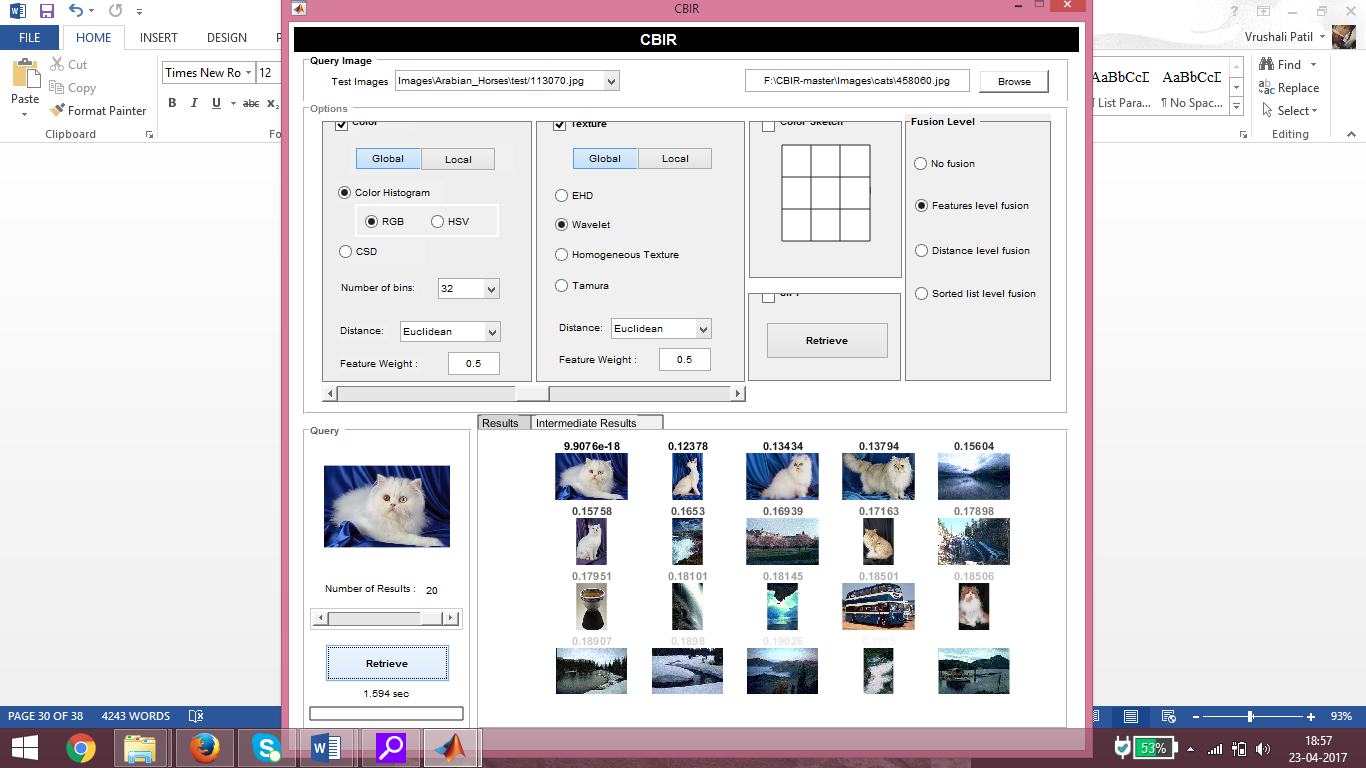


Fig.4.2.2.c.RGB and Wavelet

We use another CBIR system available by selecting RGB and Wavelet as the extraction and comparison features. The output is then analyzed and positive and negative images are counted.

Positive Images: 7/20

Negative Images: 13/20

**HSV and EHD**

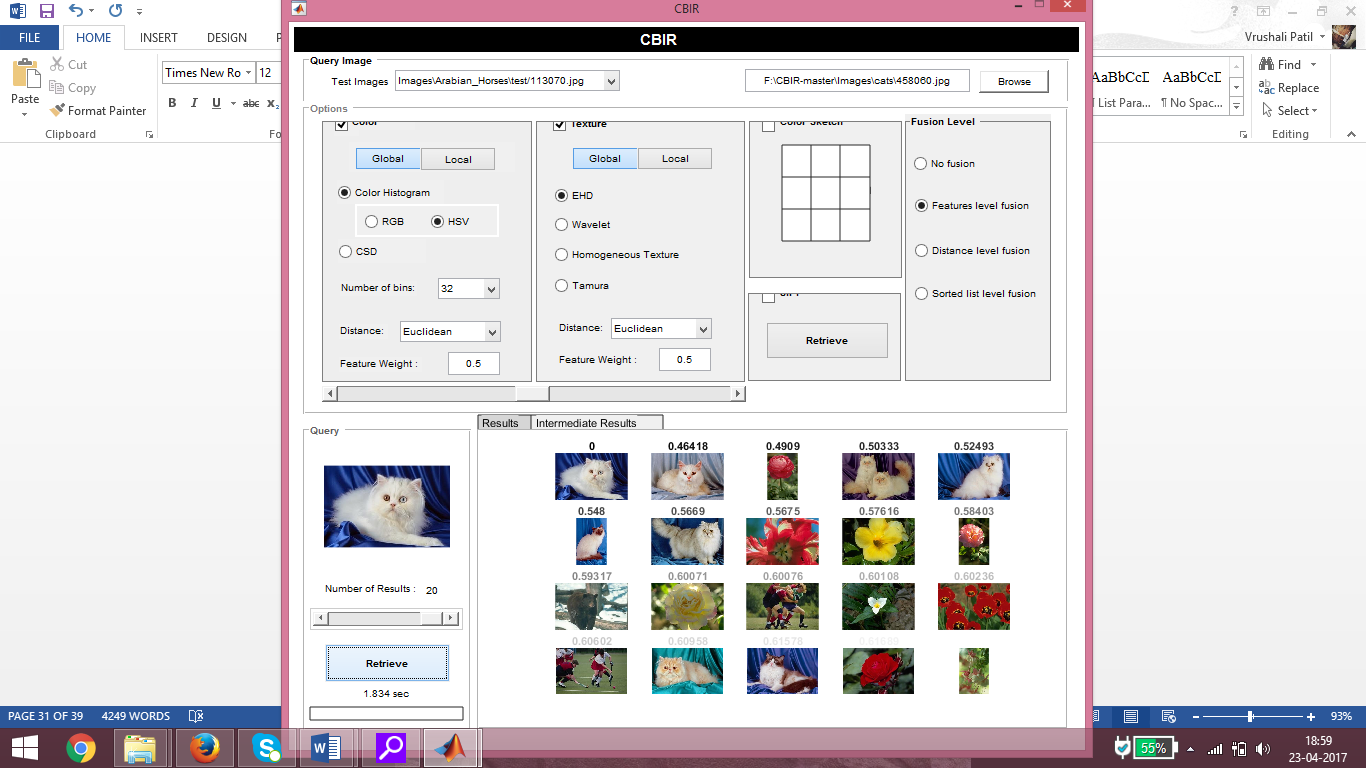


Fig.4.2.2.d.HSV and EHD

We use another CBIR system available by selecting HSV and EHD as the extraction and comparison features. The output is then analyzed and positive and negative images are counted.

Positive Images: 8/20

Negative Images: 7/20

**HSV and Wavelet**

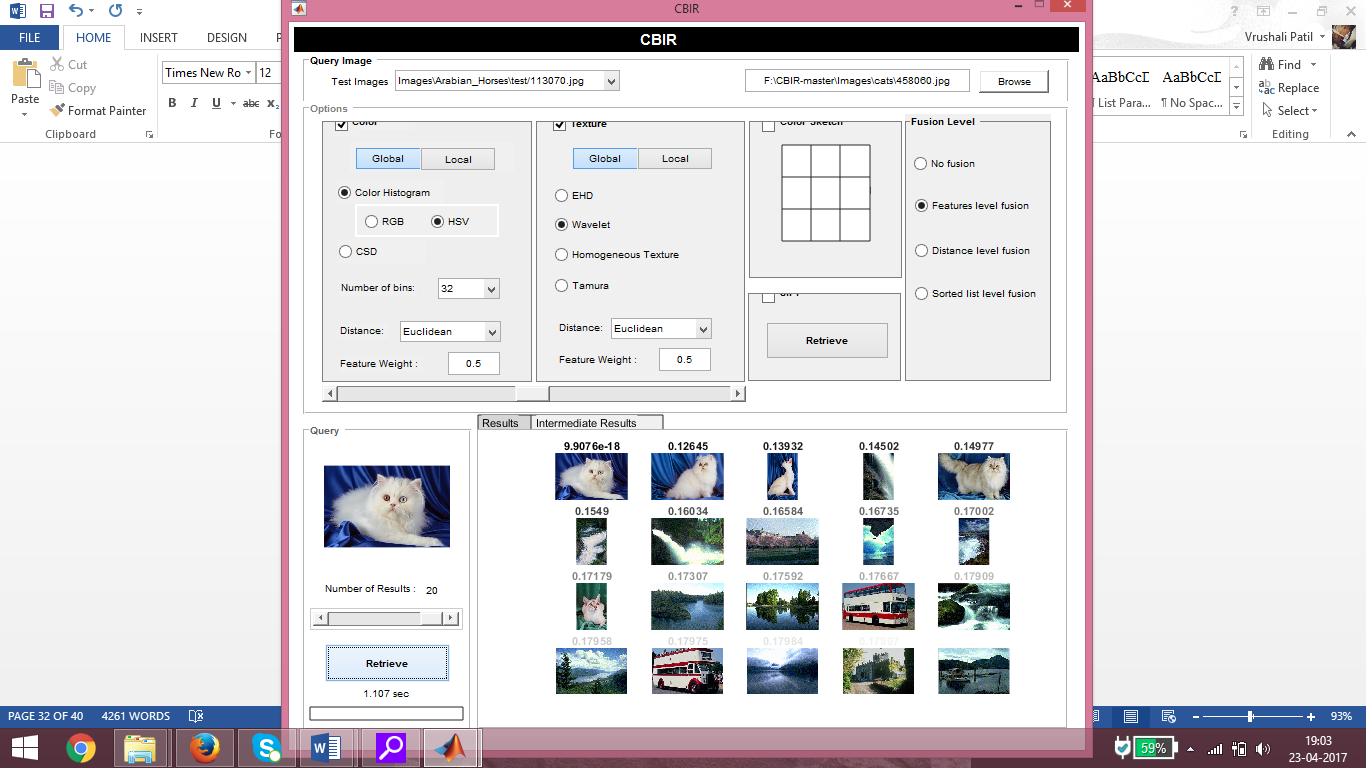


Fig.4.2.2.e.HSV and Wavelet

We use another CBIR system available by selecting HSV and Wavelet as the extraction and comparison features. The output is then analyzed and positive and negative images are counted.

Positive Images: 5/20

Negative Images: 15/20

* + 1. **Case Scenarios**

Here we will analyze the kind of query which will produce the worst, best and average case scenario and study them.

* + - 1. **Worst case scenario:**

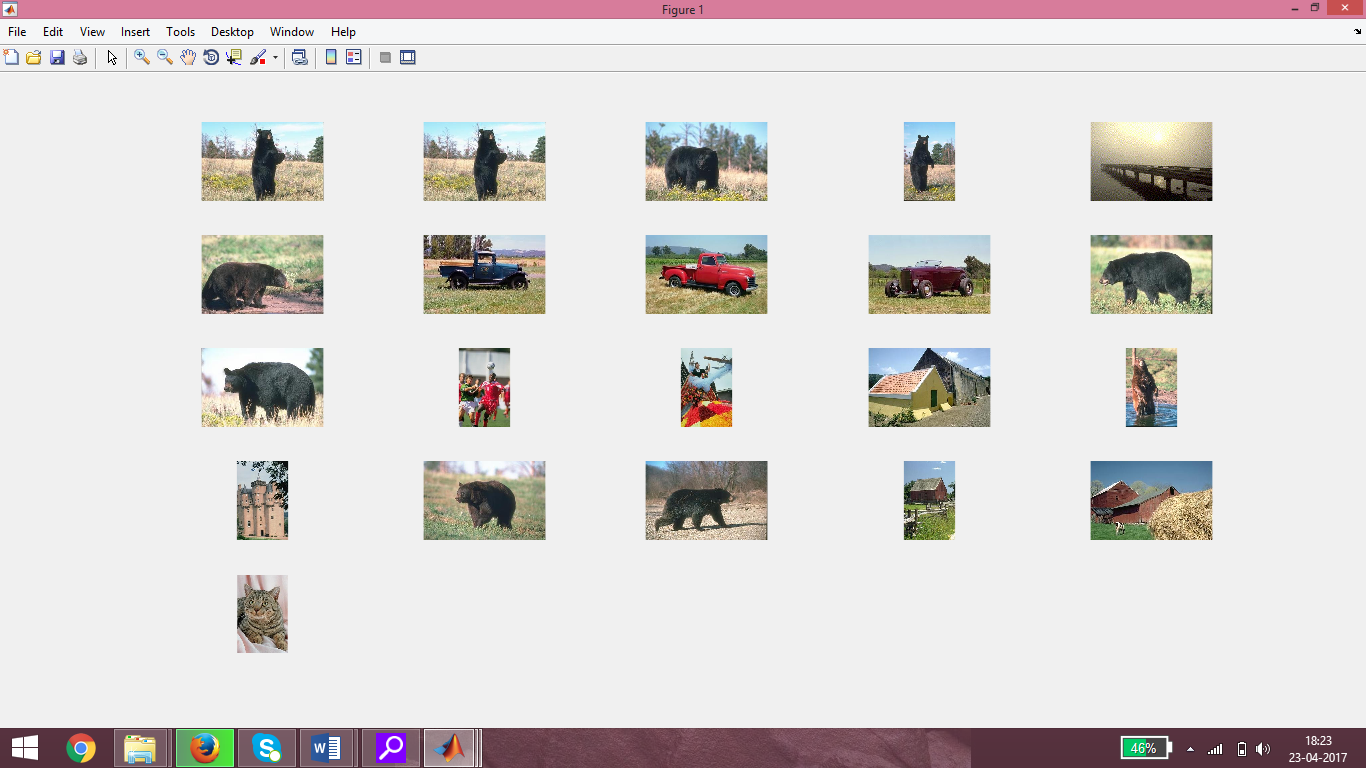


Fig.4.2.3.1.Worst case scenario

Here, the query does not have a clear distinguishable foreground and the background consists of the same color shade and covers a large portion of the image. Hence we get images which have the same background in addition to images with same foreground.

Positive images: 8/20

Negative images: 12/20

* + - 1. **Good case scenario:**

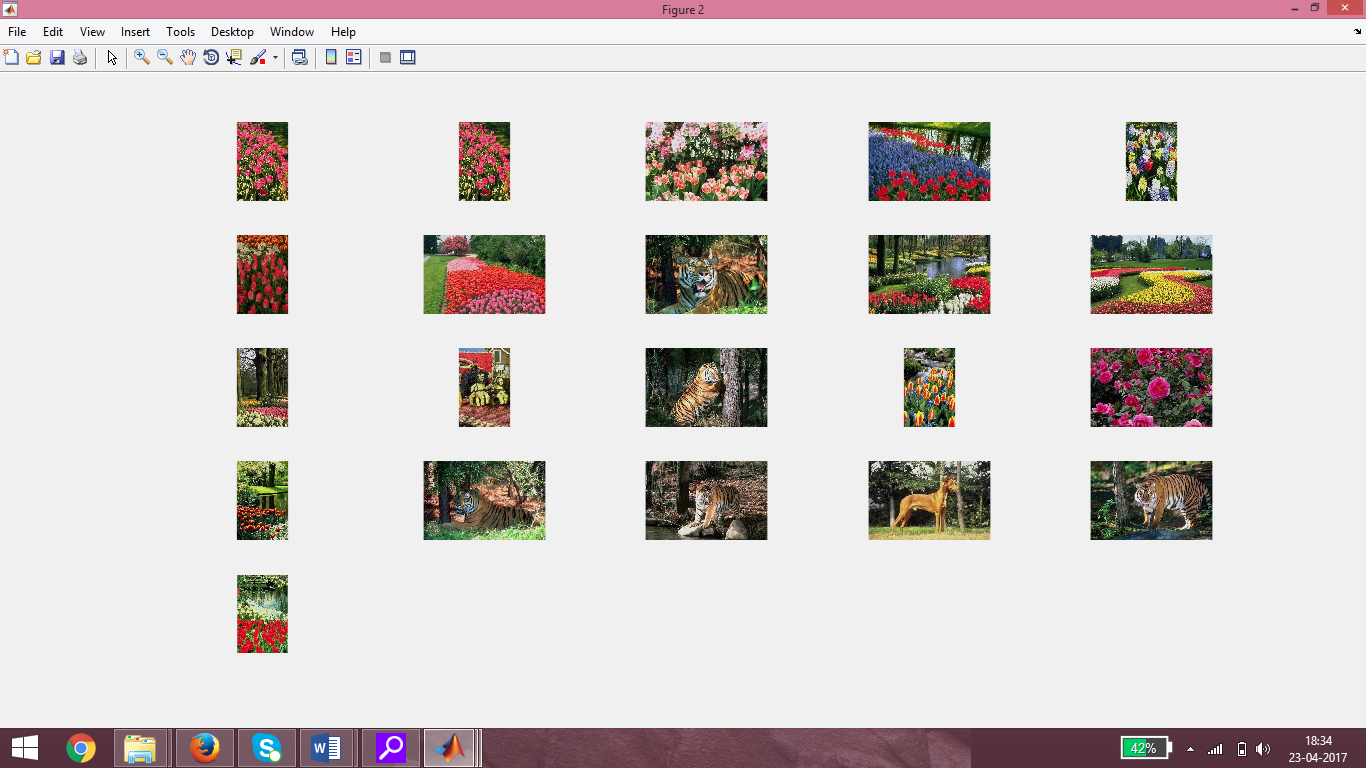


Fig.4.2.3.2.Good case scenario

Images which are not similar but have the same color combinations can have a great amount of feature similarity between them. The entire image may be considered as a negative output, but some part of it does match the query.

Positive images: 14/20

Negative images: 6/20

* + - 1. **Best case scenario:**

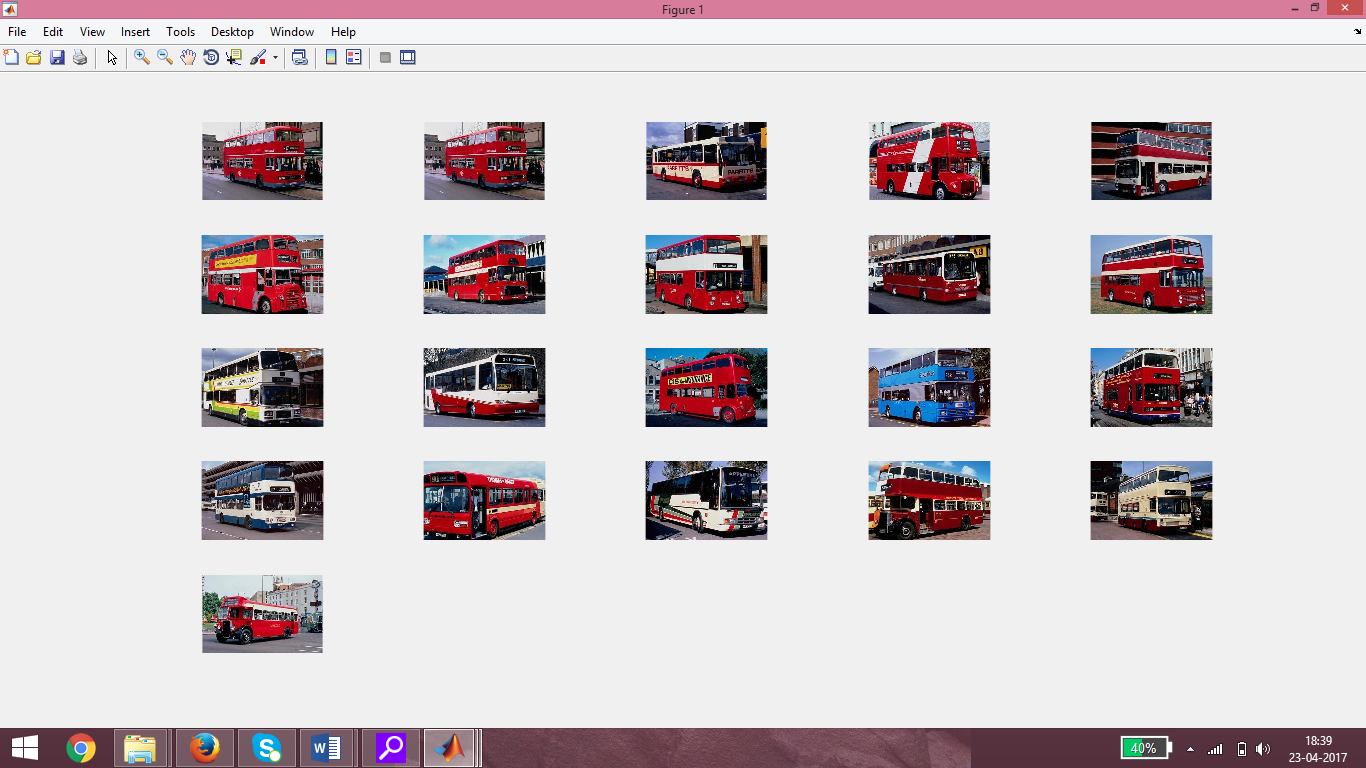


Fig.4.2.3.3.Best case scenario

The query image has a well-defined shape and also the color distribution for the entire image is well balanced. In such cases it becomes very efficient to get good outputs which are completely efficient.

Positive images: 20/20

Negative images: 0/20

* 1. **Tabular analysis**

Efficiency = Positive images/ Total images \*100

|  |  |  |  |
| --- | --- | --- | --- |
| CBIR technique | Positive images | Negative images | Efficiency |
| HMMD and EHD(cat) | 17/20 | 3/20 | 85% |
| RGB and EHD(cat) | 9/20 | 11/20 | 45% |
| RGB and Wavelet(cat) | 7/20 | 13/20 | 35% |
| HSV and EHD(cat) | 8/20 | 12/20 | 40% |
| HSV and Wavelet(cat) | 5/20 | 15/20 | 25% |
| HMMD and EHD(worst case) | 8/20 | 12/20 | 40% |
| HMMD and EHD(good case) | 14/20 | 6/20 | 70% |
| HMMD and EHD(best case) | 20/20 | 0/20 | 100% |

Fig.4.3.Tabular analysis

It can be observed that this technique is quite efficient. It performs way better than using some other existing features and in a few cases performs similar to them. It is only for a few exceptions that the performance degrades. Which should be acceptable given that image search is not as subjective as text based search.

# Chapter 5

**Future Work**

Our future effort is to add the ability to allow the user searching based on the feature of his choice in addition to the hybrid method used in the project. New functionalities that can be added in to this system. Image renaming feature can be added on the search panel. The output images could be renamed with a similar prefix or suffix in order to use the project as a sorting method to find only targeted images easily. Feedback for output could be asked to the used asking for satisfaction levels on using the system or to select the images he feels should not be shown. The user could provide feedback for the output he gets. Based on this feedback a dedicated learning algorithm can be used to improve the results after each use. Dedicated application could be built as a product for the market. There are tools available which can help convert a matlab code into an Android application. We could strive and make it into a dedicated product.

# Chapter 6 Conclusion

Color and Shape based Image Search Engineis a good way of performing CBIR with good efficiency. The system has been developed with so much care that it is free of errors and at the same time efficient and less time consuming. System is robust. The system is scalable and new filters can be added as well as new images can be added to the database.

The main purpose of system is to find out whether the features to be used decided by us are competitive enough with the current systems being used. It is concluded that the system fares better than the other extraction feature systems which are available on the internet. The HMMD features are way better than RGB and HSV when it comes to color based extraction as it is developed as a further iteration to these two. Also the EHD proves to be a better method than Wavelet and other segmentation methods. There are certain anomalies however that arise due to the combination of these two features. Too much of a color in the background takes away the weightage of the centre object. Such problems exist in every CBIR system but with further analysis we can surely work on improvement. This system is a research oriented system with several features mainly producing unique output of CBIR. The result indicates the potential evidence that HMMD and EHD can be used and improved to create an efficient Image Search.

## References

**[1]** “Content-based image retrieval using color and texture fused features”,Jun Yue, Zhebo Li, Lu Liu, Zetian Fu, Science Direct: Mathematical and Computer Modelling, August 2011.

**[2]** “Content Based Image Retrieval Using Color Histogram”, A. Ramesh Kumar, D. Saravananan, (IJCSIT) International Journal of Computer Science and Information Technologies, 2013.

**[3]** “Interactive Image Segmentation with GrabCut”**,** Bryan Anenberg, Michael Meister,Published as Anenberg papers at Stanford University.

**[4]** “Image segmentation using GrabCut”, Thomas Malmer, Published on Matematiklth, May 12, 2010.

**[5]** “Content Based Retrieval and Classification of Images Using Hybrid of HMMD Color Space”, Sheikh Aamir Rasool, Amarinder Kaur, Published in International Academy of Engineering and Medical Research, March 2017.

**[6]** “Content Based Image Retrieval using HMMD”, Harkirat Kaur, Baljit Singh Khehra, International Journal of Research, June 2016.

**[7]** **“**Improved CBIR system using Edge Histogram Descriptor (EHD) and Support Vector Machine (SVM)**”**, Pradnya Vikhar, Pravin Karde, Published in ICT in Business Industry & Government (ICTBIG), International Conference, 2016.

**[8]** https://in.mathworks.com/

**[9]** https://github.com/aminert/CBIR

## Acknowledgement

We hereby are grateful to present our B.E. Project II on **“Image Search Engine”** through this report. We would like to express our sincere respect to our Project Guide **Prof. Namita Pulgam** who helped us throughout the whole project and gave us her valuable time and advice, contributing her experience and knowledge. We thank our Honorable Principal **Dr. Ramesh Vasappanavara** for extending his full-fledged support and cooperation for development in this project. We express our gratitude towards our Head of Computer Engineering Department **Dr. Leena Ragha** for their valuable advice andsupport.

We honorably thank our entire Computer Department Lab In-Charge and assistants who have willingly co-operated with us in resolving our queries and providing us with all the required help and favor ontime.