**Ain Shams University**

**Faculty of Engineering**

**Computer & Systems Engineering Department**

Content-based Multimedia Retrieval System

A close up of a sign

Description automatically generated**submitted by :**

**ريم ناصر عبد الفتاح ممدوح عانوس 1600594**

**شيماء محمد عبد الله حسن منصور 1600703**

**عبدالرحمن عاطف محمد 1600790**

**عمر احمد عبد الله يوسف 1600851**

**دعاء هشام حسين 1600524**

**عمر حاتم عبد الشافي 1600865**

**بانسيه ياسر محمد حافظ 1600386**

**الاء محمد عبد الرحمن حسين 1600281**

# **Introduction**

This project is about Content-Based Multimedia Retrieval systems. Multimedia is a form of communication that combines different content forms such as text, audio, images, animations, or video into a single presentation, in contrast to traditional mass media, such as printed material or audio recordings. Popular examples of multimedia include video podcasts, audio slideshows and Animated videos. Multimedia can be recorded for playback on computers, laptops, smartphones, and other electronic devices, either on demand or in real time (streaming).

Multimedia system is the system that involves generation, representation, storage, transmission, search and retrieval, and delivery of multimedia information.

Developing a successful multimedia system is non-trivial because:

* Continuous media types such as video need a lot of space to store and a large bandwidth to transmit.
* They also have tight timing constraints.
* Automatically analyzing, indexing, and organizing information in audio, image, and video is much harder than text.

Multimedia database has to deal with large media files nowadays. Consider for example google or YouTube, they need to deal with hundreds of thousands of terabytes every day. So, we need to use efficient data structures, indexing and searching methods.

Researchers discovered many algorithms to use in content-based image retrieval systems, and content-based video retrieval systems. Popular algorithms in image retrieval are histogram, mean color and color layout. Popular algorithms in video retrieval systems are text-based method, metadata-based method, content-based method, audio-based method and integrated approach.

Content-Based Multimedia Retrieval systems are in strong need nowadays because we live in the big data era, where we deal with millions of gigabytes each day and we want to process these data, search, and retrieve them in an efficient way and in a short time, and this is what this project is about.   
 

Table of Contents

[**Introduction** 1](#_Toc74497591)

[**1.** **Detailed Project Description** 3](#_Toc74497592)

[**2.** **Beneficiaries of the project** 4](#_Toc74497593)

[**3.** **Detailed analysis** 4](#_Toc74497594)

[**4.** **Detailed description of the adopted techniques** 5](#_Toc74497595)

[**1.** **Color layout** 5](#_Toc74497596)

[**2.** **Color Histogram** 6](#_Toc74497597)

[**1.** **Color Histogram Similarity Measures** 6](#_Toc74497598)

[**3.** **Mean Color:** 7](#_Toc74497599)

[**4.** **Dominant Color:** 8](#_Toc74497600)

[**5.** **Video Retrieval** 8](#_Toc74497601)

[1. Histogram Based Video Retrieval 8](#_Toc74497602)

[2. Key Extraction 9](#_Toc74497603)

[3. Histogram Extraction 9](#_Toc74497604)

[4. Matching Algorithm 9](#_Toc74497605)

[**5.** **Task breakdown structure** 10](#_Toc74497606)

[**6.** **Time plan Gantt Chart** 12](#_Toc74497607)

[**7.** **System architecture** 13](#_Toc74497608)

[**8.** **Multimedia database design** 14](#_Toc74497609)

[**9.** **System design** 14](#_Toc74497610)

[**10.** **Testing scenarios and results:** 17](#_Toc74497611)

[**11.** **End User Guide** 20](#_Toc74497612)

[**12.** **Conclusion** 23](#_Toc74497613)

[13. References: 23](#_Toc74497614)

# **Detailed Project Description**

This project is about Content-Based Multimedia Retrieval systems. Multimedia retrieval systems in this project are divided into two parts: content-based image retrieval systems and content-based video retrieval systems. The purpose of this project is to provide three algorithms to retrieve images from the database, and one algorithm to retrieve videos from the database.

The user should select whether he wants to retrieve an image or a video. If he wants to retrieve an image, he should select an algorithm of the three algorithms used to retrieve the image. This project supports an algorithm for video retrieval, so the user Is forced to use it. The input then, should be an image or a video. After that the user should click Search in the Gui and the program should work to retrieve the image or the video from the database based on the algorithm the user selected earlier.

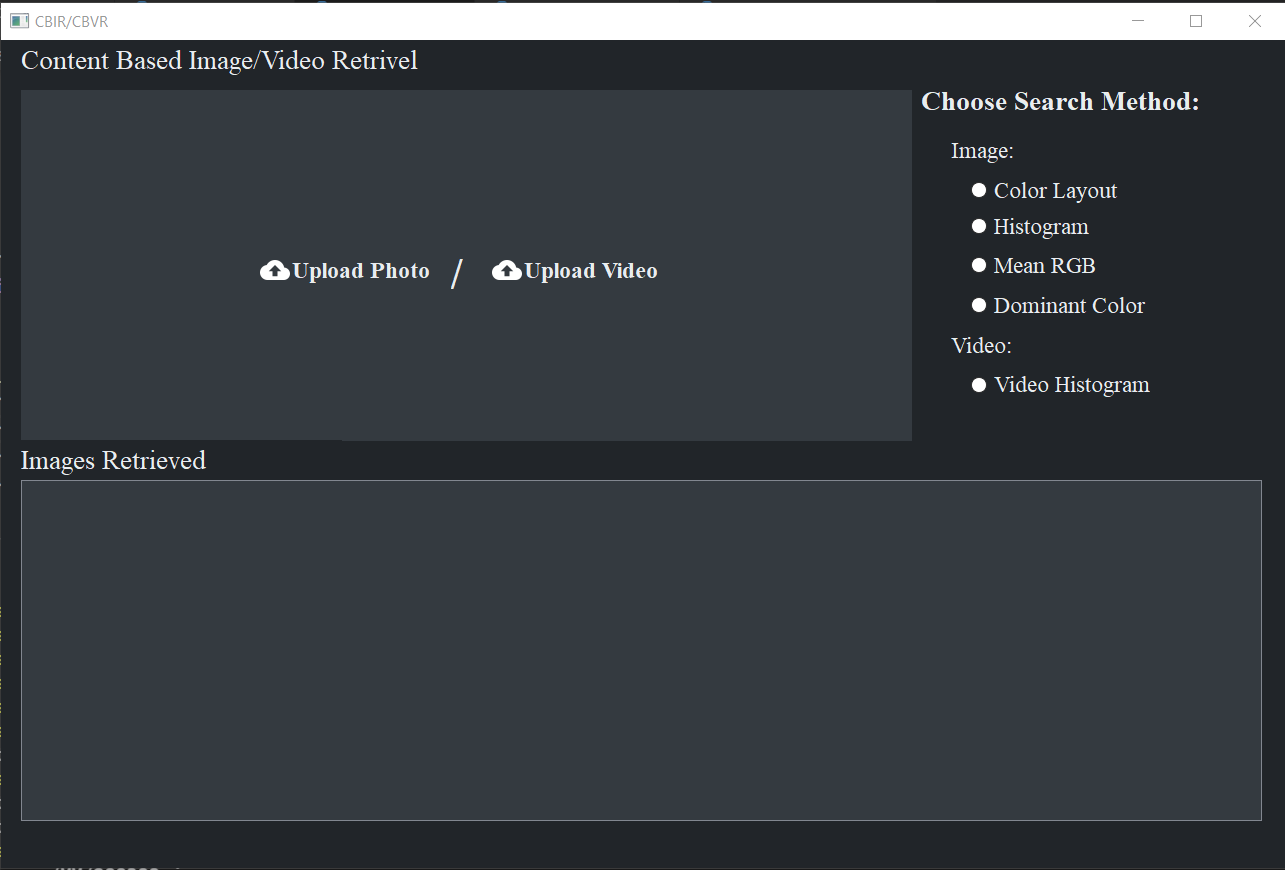
Images and videos are stored in the database in terms of their features and a link to the actual image or video itself. When the user enters an image or video the algorithm runs to convert this image into its features so that it can be comparable with the features in the database.

The output of the program should be a collection of images or a collection of videos, based on the user input, that will best match the entered image or video depending on the selected algorithm.

The project uses three algorithms for image retrieval: Color histogram, Mean color, and color layout algorithms.

It uses one algorithm for video retrieval: Histogram feature

extraction.



# **Beneficiaries of the project**

Content-based multimedia information retrieval provides new models and methods for effectively and efficiently searching through a huge variety of media that are available in different kinds of repositories (digital libraries, Web portals, social networks, multimedia databases). One way to retrieve image information is the traditional textual based method, which is manual annotation of images with text description, but it has too many limitations as it is subjective to human perception , too much responsibility on end user and some abstract needs that some images depend just on visual features of it so it is time consuming and may not capture the keywords desired to describe the image. Finding a way to extract features from images and videos in automated way without depending on human perception is much faster and more accurate. Nowadays , we have large and growing amount of digital data, and the development of the Internet highlight the need to develop efficient methods that provide more than just simple text-based query but describe the media dependent on its features only.

# **Detailed analysis**

The project is mainly divided into different phases.

**First phase** is an overview on the interface and how the user can have an easy and good experience with the app. The user can choose to upload a photo or a video to use as a query, depending on which button they click on. A popup window will appear to the user to upload the desired photo/video from the computer.

**Second phase** is which techniques to be adopted in the project . We choose 3 techniques for image retrieval (Color Histogram, Color layout, mean color, dominant color) and 1 for video retrieval (color histogram method).

**Third phase** is the design for our database to handle the features representing the images and videos.

**Fourth phase** is integrating our modules to test our app when querying some images and videos and checking the results.

# **Detailed description of the adopted techniques**

## **Color layout**

Color layout is another technique in content-based image retrieval systems. As we saw previously that when we use histogram approach on the whole image, the results are not the great, for example, Global color features (mean color) give too many false positives.

It works by dividing the whole image into sub-blocks, then extracting features (colors) from each sub-block.

The accuracy of the output of this algorithm depends strongly on the number of sub-blocks, and on how you divide these sub-blocks, for example, research said that if you divide the image into regions based on color feature concentration you will achieve great results. This process is known as segmentation. Dividing the image into a large number of sub-blocks will make accuracy better but it will be computationally expensive.

To measure the similarity between the query image and the images stored in database, you should convert the image first in its features by using this algorithm, also, you should store the images in the database by features using the same algorithm. After that, there are multiple scenarios that can occur for the user in order to query the image. The user can specify regions with specific color, or he could divide the image into a finite number of grids. Then, associate each grid with a specific color (chosen from a color palette). It is also possible to provide the information from a sample image, and that is the approach used in our code.   
Color layout measures that use a grid requires dcolor : a grid square color distance measure) that compare the grids between the sample image and the matched image (in DB).

*dgridded\_square (I, Q) = Σ dcolor (CI (g), CQ(g))*

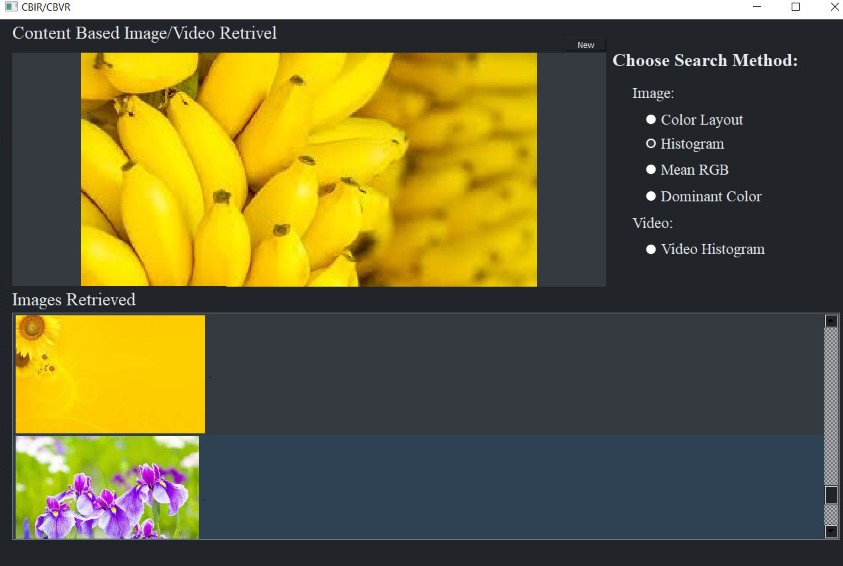
where CI(g) and CQ(g) represent the color in grid g of a database image I and query image Q respectively.

By testing we found that the color layout technique achieved better results than color histogram and mean color approaches.  

## **Color Histogram**

The color histogram is a method for describing the color content of an image, in other words it is a primitive features extraction method, it is commonly used color feature representation. it counts the number of occurrences of each color in an image. The color histogram of an image is rotation, translation, and scale-invariant; therefore, it is very suitable for color-based CBIR using global color features of images. However, the main drawback of their use is that they do not take account of the information represented by the color.

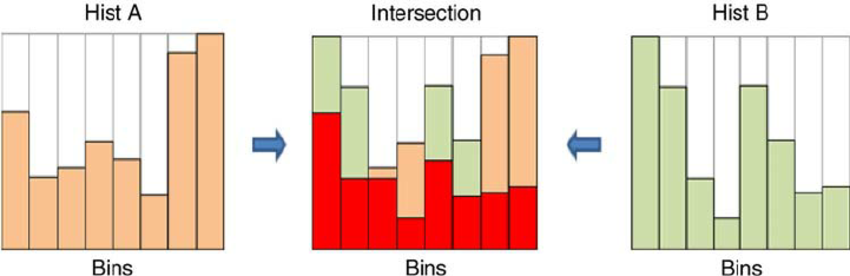
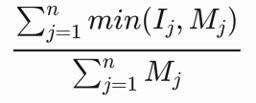
***Example for drawback***

****

First, we calculate the histogram for all the images dataset that will be save in our database in HSV color model (hue, saturation, value), This representation is much closer to human eye perception of color so we use it in our code.

Second, we normalize the histogram to be independent of images size then we save it in the database.

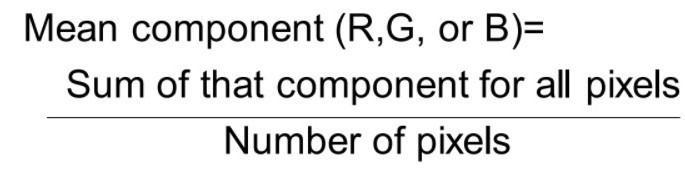
### **Color Histogram Similarity Measures**



Color histogram distance is the sum of the smallest bin for each intersecting bins in the two histograms for input query image **I** and the model database image **M** normalized divided by the

number of pixels in the database image. If the distance between the 2 images is less than the defined threshold (threshold =10^-5) so this image will be retrieved.

## **Mean Color:**



Mean color is a method to describe the image by its mean of colors, it is calculated by summing the red component for all the pixels and the divided this by the number of pixels, doing the same thing with the green and blue components the result is an array of three channels describing the amount of that component in the image.

In the implementation we used the built-in function **CV2.mean()** to calculate the mean color directly for both the query image and the images in our database.

## **Dominant Color:**

Dominant color is a method to describe the image by the color that is dominant all over the image which is the most spread-out color in the image.

In the implementation of this function, we used the **np.unique()** function to get the unique values from the image after converting it to array, then **argmax()**function was used to get the max values from those sorted non-repeated values of colors in the image.

## **Video Retrieval**

Finding and retrieving relevant videos from video collections is a naturally important problem. It is more necessary when videos are generated at an increasing rate nowadays, so video management and retrieval attracted a lot of researchers and a number of research has been carried out during the past decade.

Generally, video retrieval techniques are classified as concept based and content-based techniques. Since, our proposed method is a content-based technique, our current discussion is concentrated only on content-based retrieval technique.

### Histogram Based Video Retrieval

Histogram feature extraction is the most used for color feature and when applied on image it calculates the frequency count of each individual color like (R, G, B).

Video is considered as multiple images connected with each other, so we could not recognize the difference between frames if the rate of change is high enough.

Then to apply histogram extraction on video, first we should extract the key frames (frames which are different from the reference frame by a certain threshold) then calculate the histogram for every key frame.

### Key Extraction

We consider frames one, three and five frames as reference frames and then calculate the difference between the values of current frame and previous frames then divide the difference by the maximum value of current frame and previous frame, and if the difference is less than 0.1 (threshold) then current frame isn’t considered as key frame.

### Histogram Extraction

After calculating the histogram of each image using cv2.calcHist in open, then we have three arrays of histograms for every keyframe.

Then we calculate the average of histograms for all key frames by summing the values of each bin for all frames and divide it by the number of key frames, getting three histogram arrays for every video.

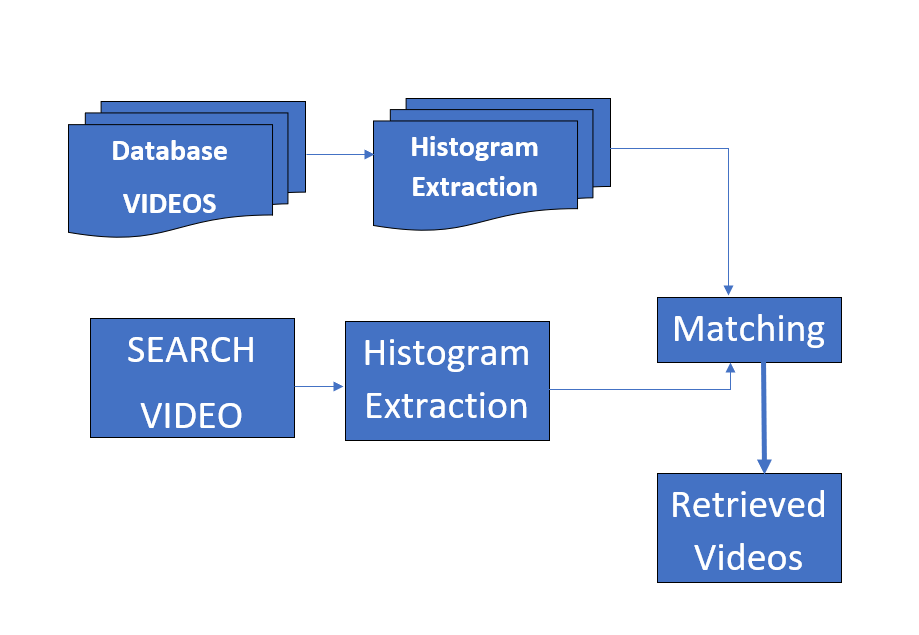
Next, we normalize the average histogram by cv2.normalize in OpenCV python to get three normalized arrays ready for matching.

Finally, save the data to a database table containing (name of video, red, green, blue) histograms.

### Matching Algorithm

After extracting the histogram features from the videos which we want to compare with, we iterate the database and calculate the difference between videos using cv2.compareHist in OpenCV python, compareHist takes three arguments (histogram of the test video, histogram of the database video, and method of comparing)

There are four methods commonly used CORRELATION and INTERSECTION which high score means better match, CHI-SQUARE and HELLINGER which low score means better match.



# **Task breakdown structure**

The project is divided into five main tasks:

*Gui Design*

* Design the frontend of the program using PyQt.
* Implement basic functionality of uploading image, uploading video.
* Implement functionality of displaying the output images or output videos.

*Algorithms Implementation*

* Implement color histogram algorithm for image retrieval and its evaluation function.
* Implement mean color algorithm for image retrieval and its evaluation function.
* Implement color layout algorithm for image retrieval and its evaluation function.
* Implement color Histogram feature extraction for video retrieval and its evaluation function.

*Database design*

* Design the outer layout of the database for images and videos.
* Convert images, and videos dataset to features and store them in the database. (MySQL database)
* Implement abstract functions to query images and videos.

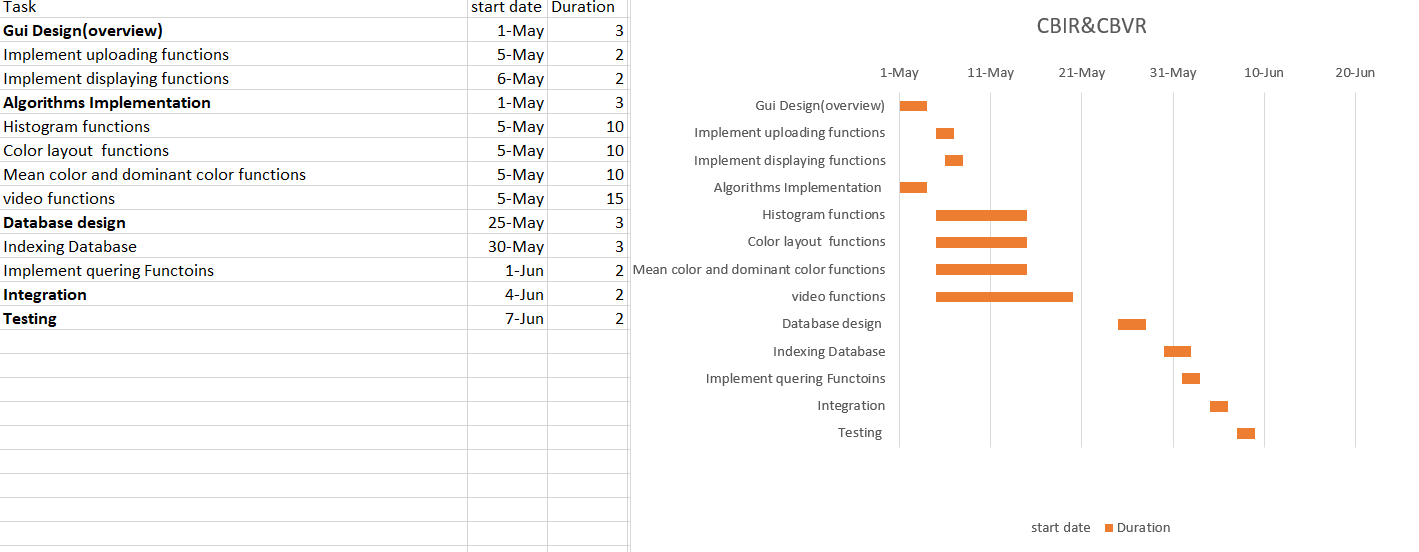
*Integration*

* Integrate all the components together to make the program is coherent.

*Testing*

* Test Gui main functionality and design
* Test three algorithms of image retrieval and the algorithm for video retrieval
* Test database query functions and indexing.
* Check for bugs and errors in the integration process between all the structures of the project.

# **Time plan Gantt Chart**



 This Gantt chart was made on excel. All tasks are from the task breakdown structure , tasks in bold were the task main phase which is the most important one.

Each task is given duration in days.

# **System architecture**

**UI**



**Insert Database image/video**

**Query image/video**



**HistogramNormalizedDiff**

**colorLayoutDifference**

**meanColorDifferenceDominant retrieve\_video**

**Histogram**

**Color layout**

**Mean color**

**Dominant color**

**Video histogram**



**Feature Extraction**





**Evaluation**

**Indexing mechanism Database**



**Similarity matching**



**Database**

**MySQL**

**Image/video Retrieval**



**UI**

 We have multiple modules in our project. The UI module handles inserting and querying image/video data. All these data features are saved in the database module. Feature extraction modules contain multiple functions to let the user choose what he needs the results to be filtered by. Filtration is done using the different function in the evaluation module which gets the similarity measure between the query and the features saved in the database.

# **Multimedia database design**

A picture containing graphical user interface

Description automatically generated

ImagesDB table:

**ID:**is the primary key and auto-incremented

Histogram: histogram array feature for model image

Colorlayout: Color layout array feature for model image

MeanColor: Mean Color array feature for model image

DominantColor: Dominant Color array feature for model image

Path: path for image in file system

VideosDB table:

**ID:**is the primary key and auto-incremented

Red: array feature for red color in keyframe

Green: array feature for green color in keyframe

Blue: array feature for bluecolor in keyframe

Path: path for image in file system

# **System design**

1. **UI module :** 2 buttons for uploading image and video for selecting a certain image/video to upload or to put in a certain directory. Once a user selects a certain image or video , if it is an image, it will be displayed , if the video it displays a popup window, and you can play the video. 5 push buttons for each retrieval algorithm. For images retrieval there is a box which displays all retrieved images and for videos the name of all videos is displayed, and you can select which one to play.
2. **Evaluation module:** This class contains the main similarity measure for each feature extraction method.
   1. **HistogramNormalizedDifference function** is for calculating the error distance when using color Histogram method, it takes both feature vectors for both the input image and the query image and get the sum of intersection between their histogram bins .
   2. **colorLayoutDifference function** is for calculating the error distance when using color layout method , the main difference between it and the histogramNormalizedDifference is that it is divided into grids.
   3. **meanColorDifference function** is for calculating the error distance when using the mean color method, getting the sub of the absolute difference between the two feature vectors.
   4. **matching the test video**: This module is used to get the best matches for the test video. First, we calculate the test video histogram using the previous module, then we iterate over all videos in the database, and calculate the difference between test video histogram and current video histogram. As a design choice, we only consider videos with similarity values greater than 0.4

inputs: test video

output: five best matched videos

1. **Feature extractor module:** This class contains the main similarity measure for each feature extraction method.
   1. **Histogram function :** calculate the histogram for each image in the database and also the query image. images are changed from BGR to HSV then resizing the image to be independent on its size, normalization for bins for more accuracy.
   2. **ColorLayout function :** image is divided into regions (grid size) to calculate the histogram for each grid in the image. images are changed from BGR to HSV then resizing the image to be independent on its size, normalization for bins for more accuracy.
   3. **generating the video histogram:** This module is responsible for taking each video in our database, and then generating an average histogram for it. Then, later on , we use this module to generate the histograms for test videos.

Inputs: video path

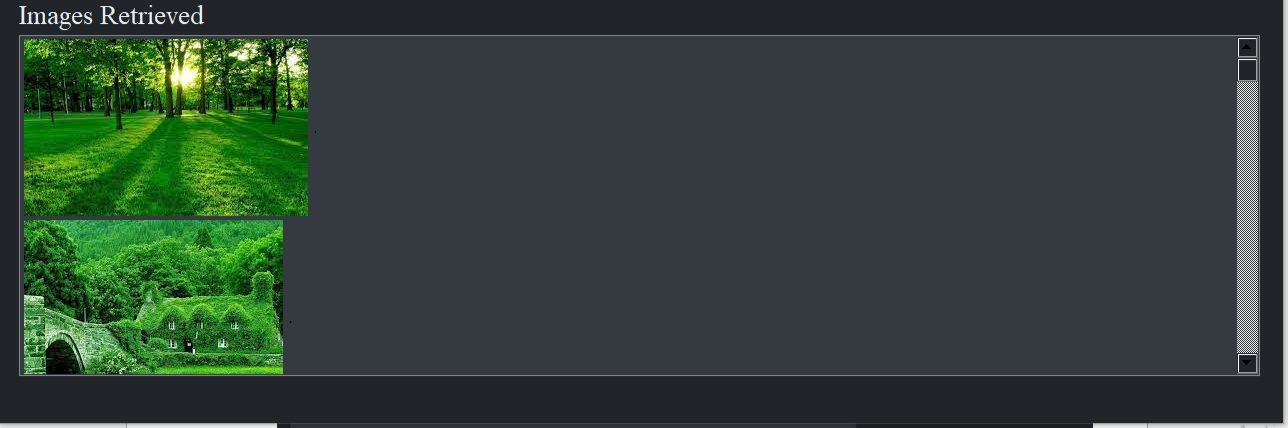
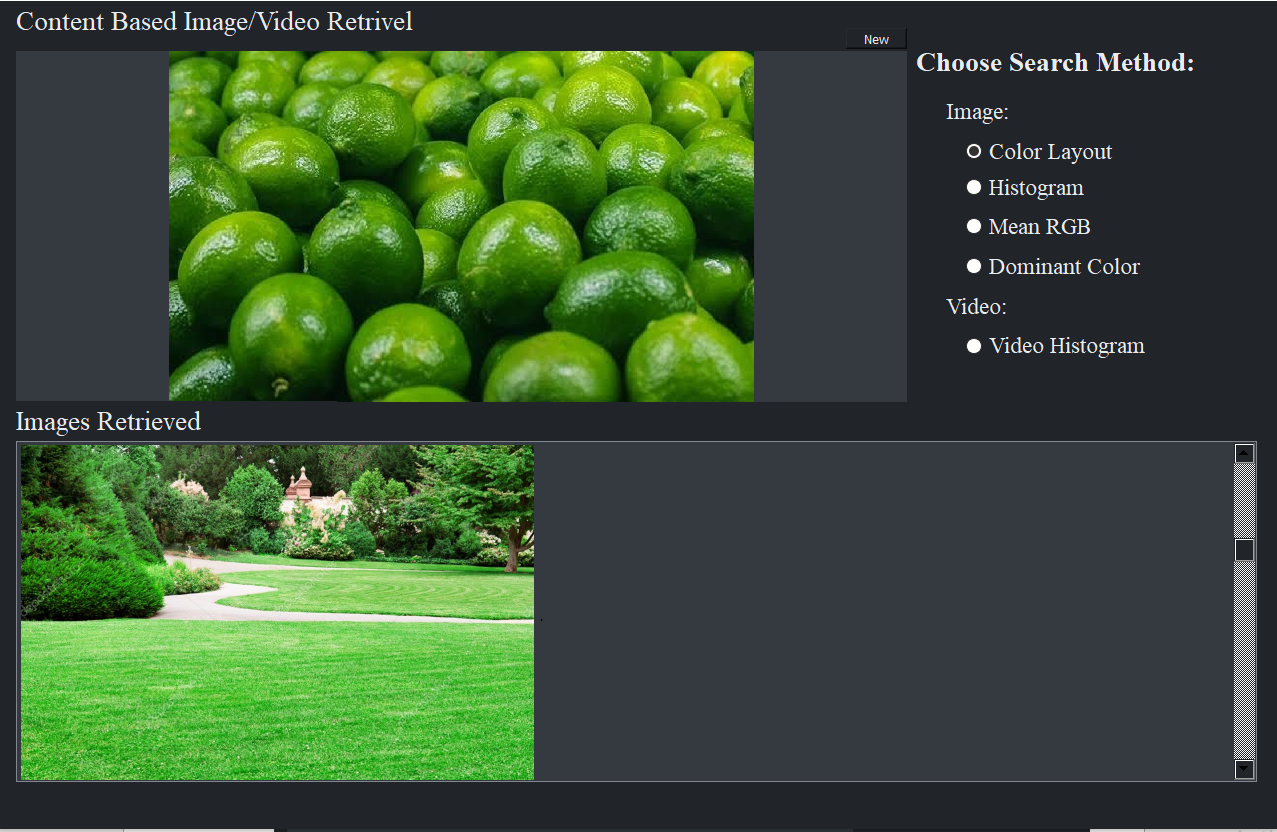
output: (3 X 255) matrix representing the video histogram

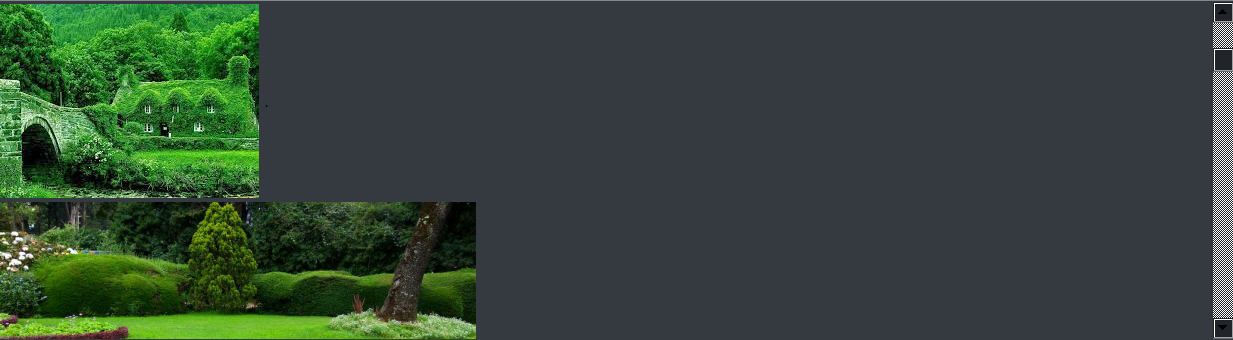
* 1. **mean\_color :** calculate the mean color for both the images in the database and the query image, calculating the mean color is done by computing the amount of each of the Red , Green and Blue component in the whole image and then divided this by the total number of pixels of that component resulting an array of three channels describing each one of those RGB components.
  2. **Dominant\_color :** calculate the dominant color in the image by getting all of the unique values of colors in the image and getting the number of pixels with each of these colors throughout the image and then getting the max of these values representing the most spread color all over the image.

1. **DB module:** it is the main database class for creating the database for the first time indexing the features in it .Also for creating tables for images and videos features to save with it the path of media in the file system. Also, for retrieving the best matching media from the database. so, it is all about dealing and accessing the tables saved in DB.
2. **CBIR module:** main function is to save the features for images and videos in the database for the first time.

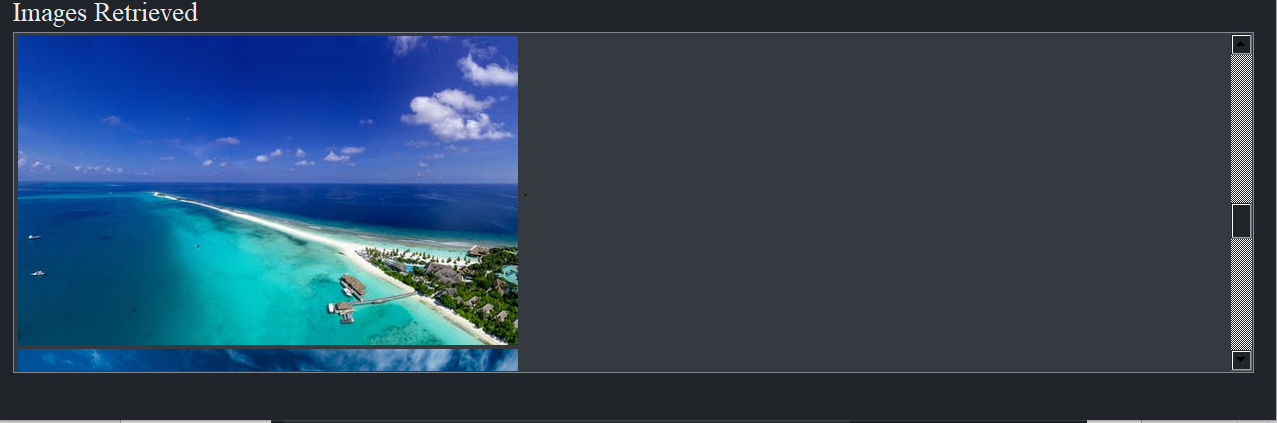
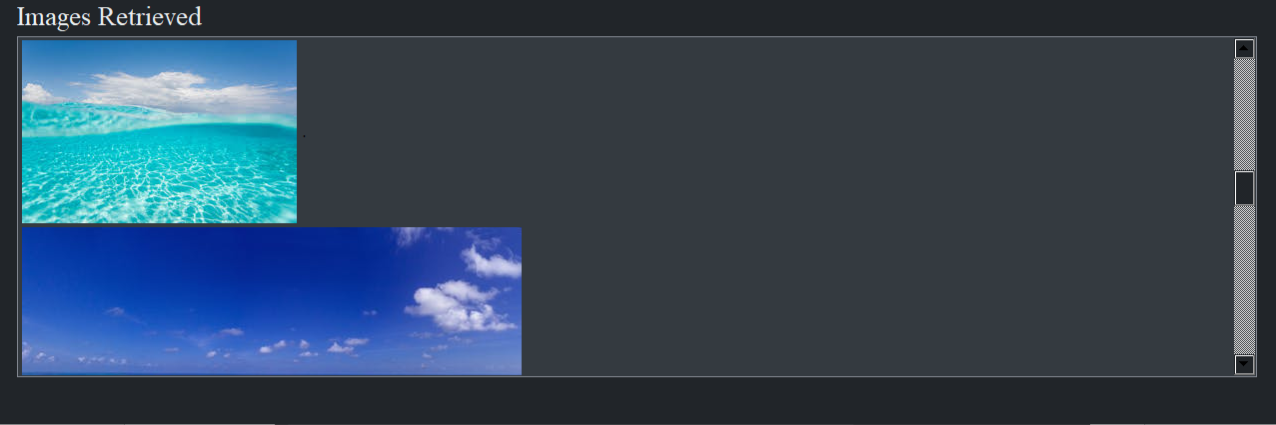
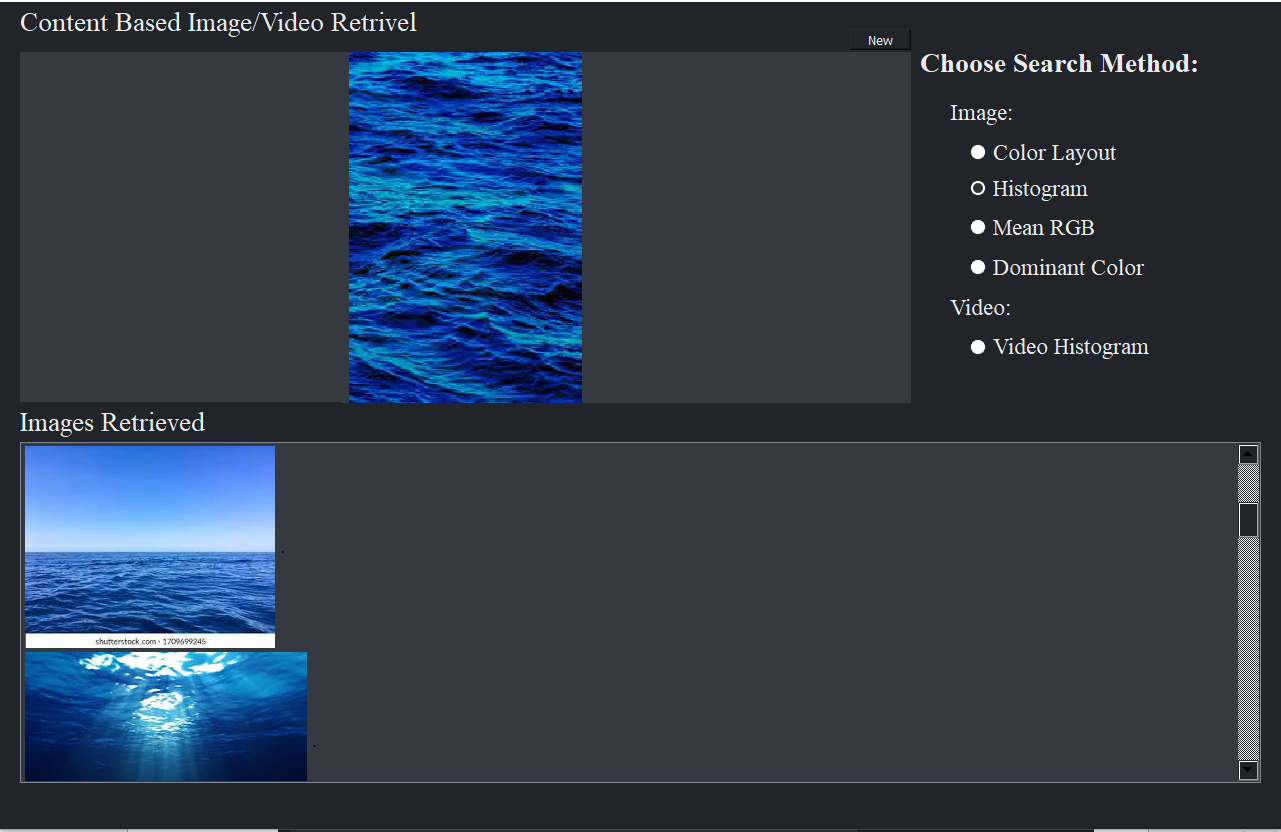
# **Testing scenarios and results:**

Color layout example:



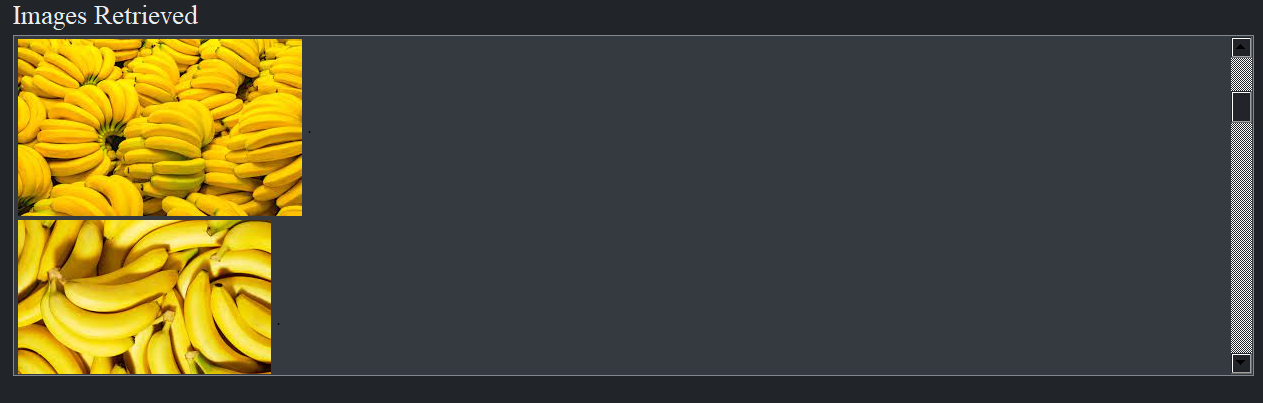


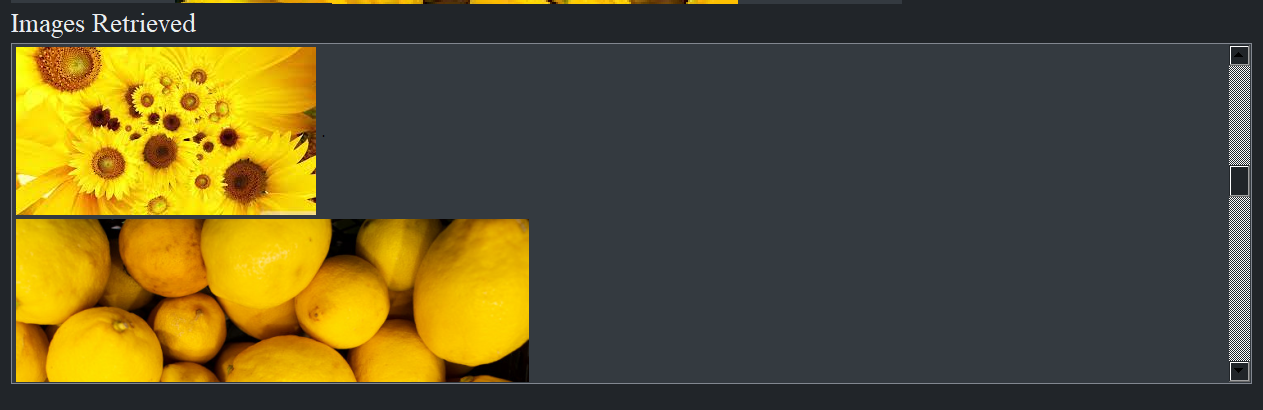
Histogram example:



  Mean RGB example:

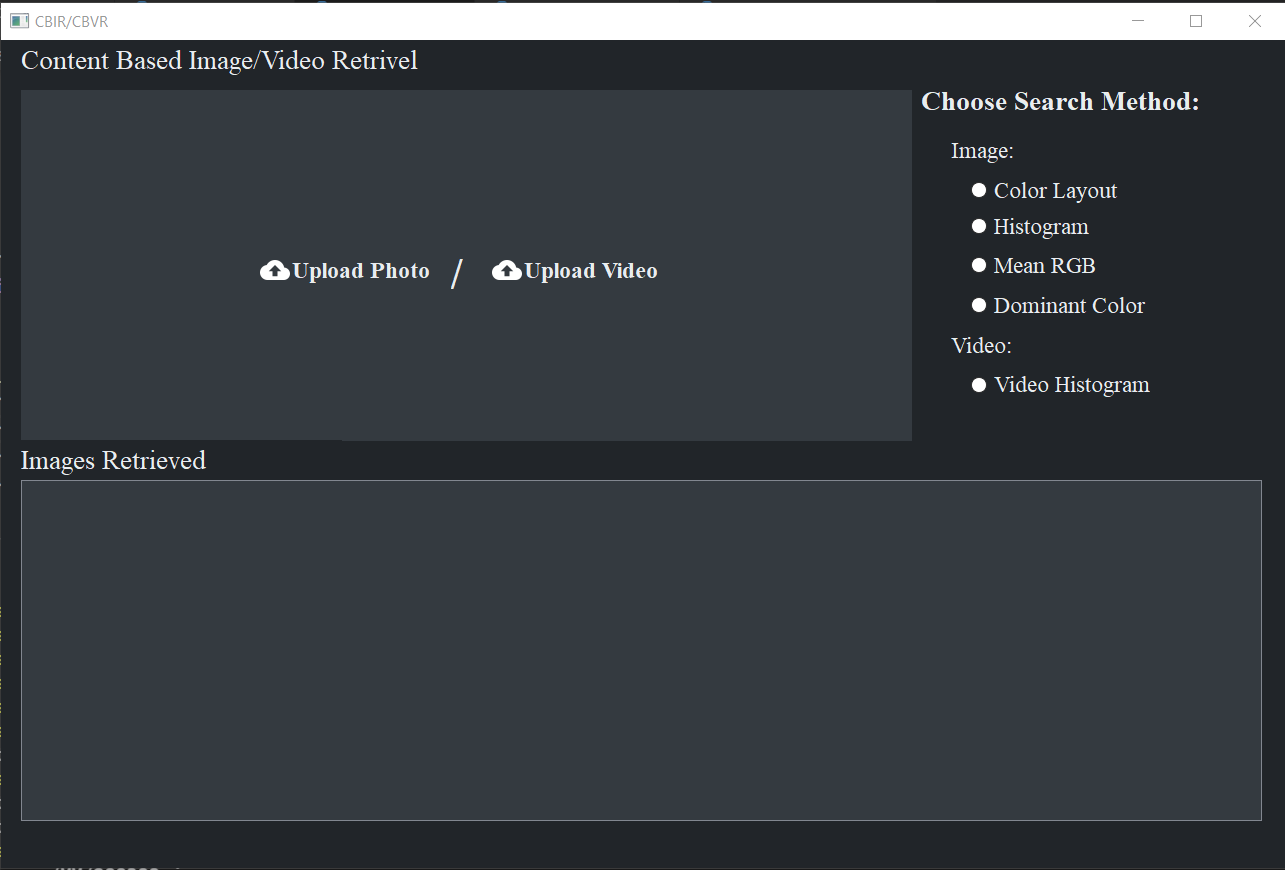




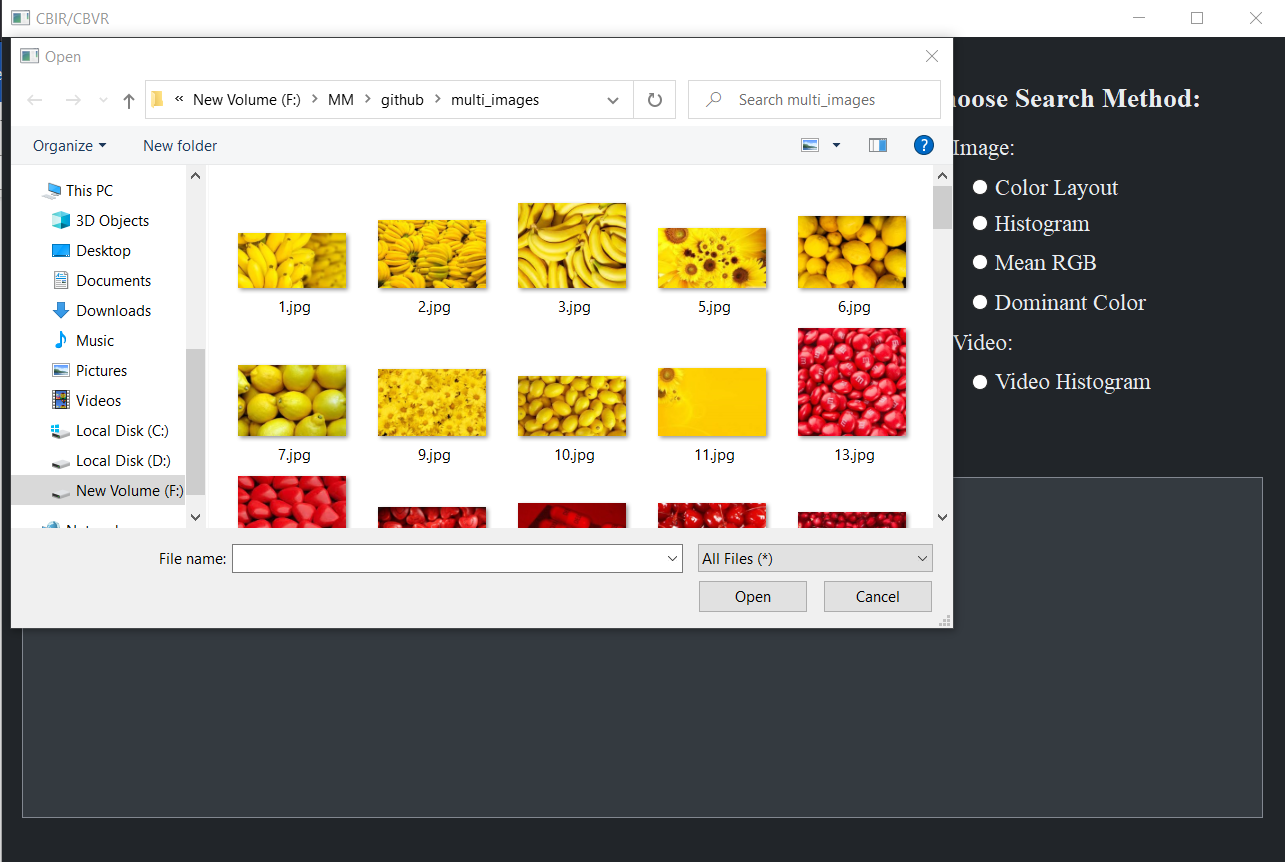


# **End User Guide**

The user interface of the application is quite simple and straightforward.

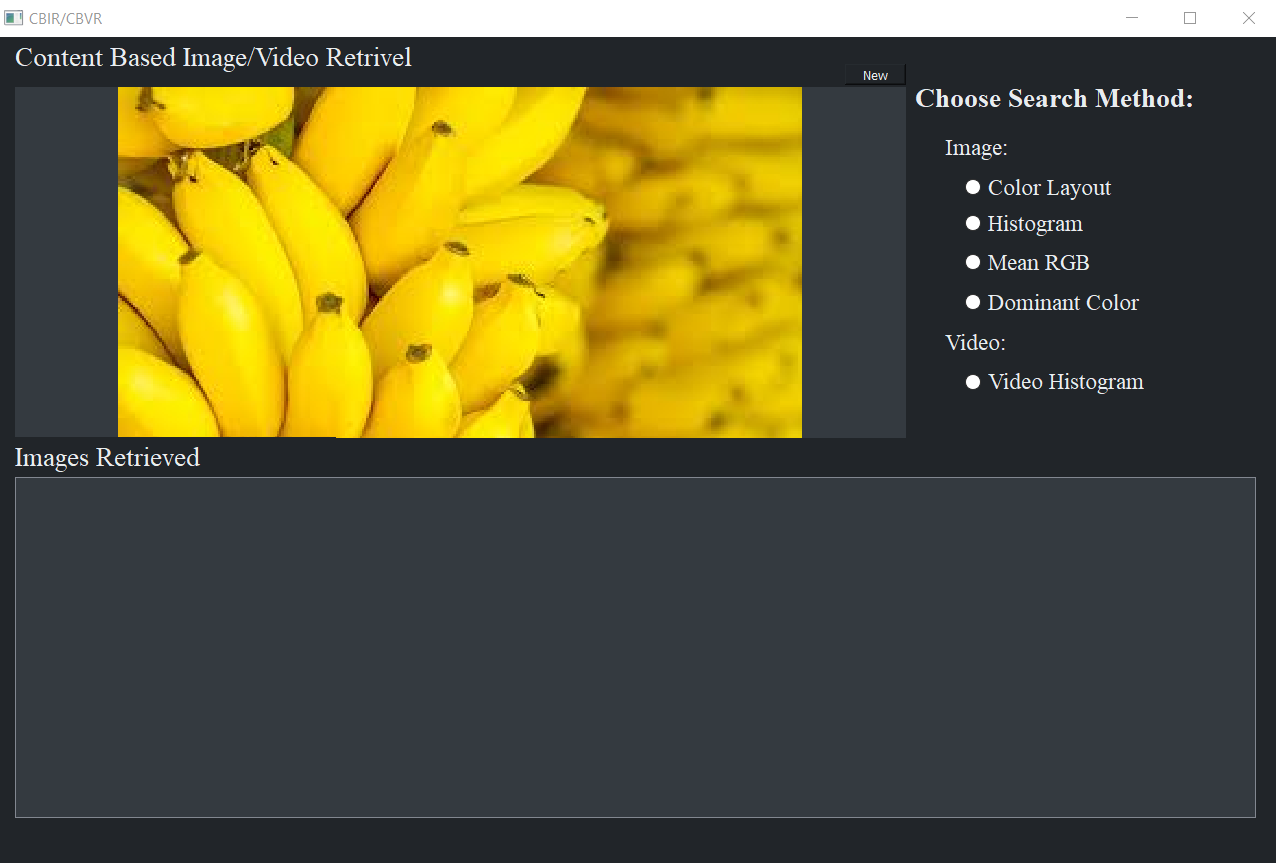


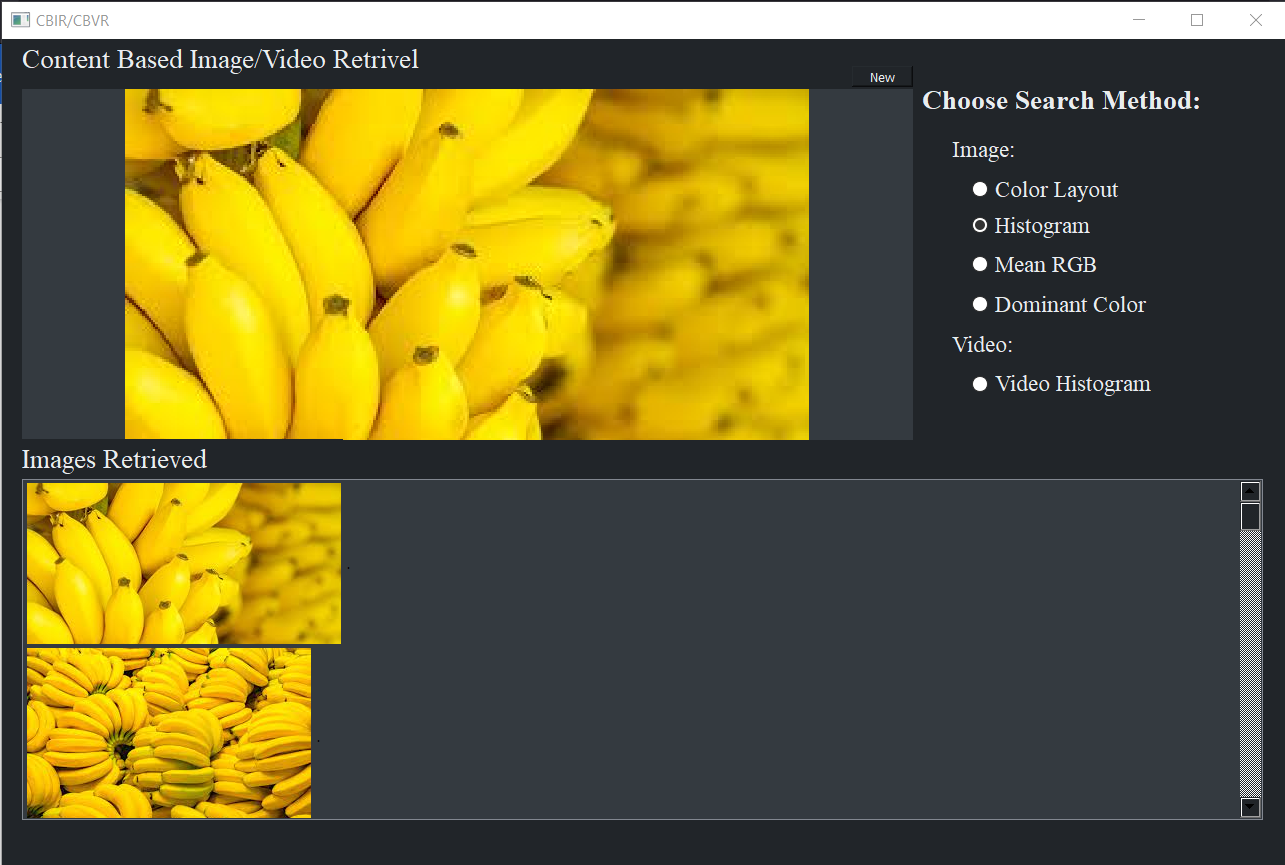
The user can choose to upload a photo or a video to use as query, depending on which button they click on. A popup window will appear to the user to upload the desired photo/video from the computer.



The chosen picture is displayed, and the user will then choose which retrieving method they would like to use. At this point,” New” button will appear that gives the user the ability to choose a new query at any time, by sending the user to the home page.

Once a method of retrieval is chosen, the program starts to compare the query image with the images in the database and a list of similar pictures appear in space below and you can scroll through them.





# **Conclusion**

The importance of CBIR & CBVR was shown by experiment after trying different techniques to extract features, that each one has its strength and weakness. For example, color histogram is very fast and more efficient than other techniques but can retrieve two images with different colors and act as if they are identical. Another example for mean color technique is easy to implement bit it is non-uniform as it depends on the average of the 3 channel R,G,B. All of these techniques were just extracting primitive features which are low-level features but If we use semantic or domain specific features as they are more low-level features, we will achieve good robust results.

# References:

1. <https://www.researchgate.net/publication/221551590_Color_Quantization_and_its_Impact_on_Color_Histogram_Based_Image_Retrieval>