



Faculty of Engineering & Technology – Electrical & Computer  
Engineering Department

First Semester 2023 – 2024

Computer Vision – ENCS5343

Assignment #2 – Image Retrieval System

---

Prepared By:  
Ahmaide Awawda – 1190823

Instructor:  
Mr. Aziz Qaroush

Section: 2  
Date: January 2024

## Abstract

This assignment goes on building a content-based image retrieval system (BCIR system) that retrieves the contents of similar images by extracting their features and gives feedback and outcomes based on the measurements of their similarities., with feature extraction methods such as color histograms, color moments, and more. The system will also be evaluated to measure the accuracy of its outcome at different measurements.

This assignment was implemented using the python language and a dataset from the Wang website, using OpenCV.

## ❖ Table of Content

1. Introduction.....	1
1.1. The Concept of Content-Based Image Retrieval Systems.....	1
1.2. Examples of CBIR System Implementations .....	2
1.2.1. Color Histograms.....	2
1.2.2. Color Moments.....	2
1.2.3. Other Ways of Implementations.....	2
2. System Implementation & Experimental Setup.....	3
2.1. The Used Dataset .....	3
2.2. The Used Libraries.....	3
2.3. Image Loading Function .....	4
2.4. Choosing The Queries .....	4
2.5. Feature Extraction Functions .....	4
2.5.1. Color Histogram .....	4
2.5.2. Color Moments.....	4
2.5.3. Local Binary Patterns .....	5
2.6. Feature Distance Calculation .....	5
2.7. Retrieve Images for a Given Threshold.....	5
2.8. System Evaluation .....	5
2.9. Finding Max Threshold.....	6
2.10. Receiver Operating Characteristic Curve .....	6
2.11. Plotting the Results .....	6
2.12. Implementations Values .....	7
2.12.1. Color Histogram Implementation .....	7

2.12.2. Color Moments Implementation.....	7
2.12.2. Local Binary Patterns Implementation .....	7
<b>3. Results .....</b>	<b>8</b>
3.1. Results for Color Histogram Implementation .....	8
3.1.1. Color Histogram with 120 Bins.....	8
3.1.2. Color Histogram with 180 Bins.....	9
3.1.3. Color Histogram with 256 Bins.....	11
3.1.4. Color Histogram Discussion.....	13
3.2. Results from Color Moments Implementation .....	14
3.2.1. Color Moments Equal Weights .....	14
3.2.2. Color Moments Mean Weight Double STD & Skewness .....	15
3.2.3. Color Moments STD Weight Double Mean & Skewness .....	17
3.2.4. Color Moments Skewness Weight Double Mean & STD .....	19
3.2.5. Color Moments with Extended Features .....	21
3.2.6. Color Moments Discussion .....	23
3.3. Results from Local Binary Patterns .....	24
3.3.1. LBP With 24 Points.....	24
3.3.2. LBP Discussion .....	25
<b>4. Conclusion .....</b>	<b>26</b>
<b>5. References.....</b>	<b>27</b>
<b>6. Appendix.....</b>	<b>28</b>

## List of Figures

Figure 1-1: CBIR Implementation Diagram.....	1
Figure 2-1: Query Images.....	4
Figure 3-1: Color Histogram 120 Bins Similar Images .....	9
Figure 3-2: ROC Curve for 120 Pins Color Histogram .....	9
Figure 3-3: Color Histogram 120 Bins Similar Images .....	10
Figure 3-4: ROC Curve for 180 Pins Color Histogram .....	11
Figure 3-5: Color Histogram 256 Bins Similar Images .....	12
Figure 3-6: ROC Curve for 256 Pins Color Histogram .....	13
Figure 3-7: Color Moments Equal Weights Similar Images .....	15
Figure 3-8: ROC Curve for Color Moments Equal Weights .....	15
Figure 3-9: Color Moments Double Mean Weights Similar Images .....	16
Figure 3-10: ROC Curve for Color Moments Double Mean Weights.....	17
Figure 3-11: Color Moments Double STD Weights Similar Images.....	18
Figure 3-12: ROC Curve for Color Moments Double STD Weights .....	19
Figure 3-13: Color Moments Double Skewness Weights Similar Images .....	20
Figure 3-14: ROC Curve for Color Moments Double Skewness Weights .....	21
Figure 3-15: Color Moments Extended Features Similar Images .....	22
Figure 3-16: ROC Curve for Color Moments Extended Features .....	23
Figure 3-17: LBP Extended Features Similar Images .....	25
Figure 3-18: ROC Curve for LBP Extended Features .....	25

## List of Tables

Table 2-1: Wang Dataset Categories .....	3
Table 2-2: Weights Values .....	7
Table 3-1: Color Histogram 120 Bins Retrieved Metrics .....	9
Table 3-2: Color Histogram 180 Bins Retrieved Metrics .....	11
Table 3-3: Color Histogram 256 Bins Retrieved Metrics .....	12
Table 3-4: Color Moments Equal Weights Retrieved Metrics.....	15
Table 3-5: Color Moments Double Mean Weights Retrieved Metrics .....	17
Table 3-6: Color Moments Double STD Weights Retrieved Metrics.....	18
Table 3-7: Color Moments Double Skewness Weights Retrieved Metrics .....	20
Table 3-8: Color Moments Double Extended Features Retrieved Metrics.....	22
Table 3-9: LBP Features Retrieved Metrics .....	25

# 1. Introduction

## 1.1. The Concept of Content-Based Image Retrieval Systems

In the field of digital image processing and management, Content-Based Image Retrieval (CBIR) systems are a revolutionary approach. CBIR works on extracting visual content directly from the images, as opposed to traditional retrieval techniques that depend on text-based annotations or tags. Utilizing features like color, texture, shape, and spatial layout, this methodology makes it possible to retrieve images. The fundamental idea behind CBIR is that images can be found and returned using only their intrinsic visual properties, which makes the process more natural and in line with how people see images. Digital libraries, medical imaging, social media, commerce, and criminal identification are just a few of the many uses for this technology as figure 1-1 shows a diagram of how the system works. [1]

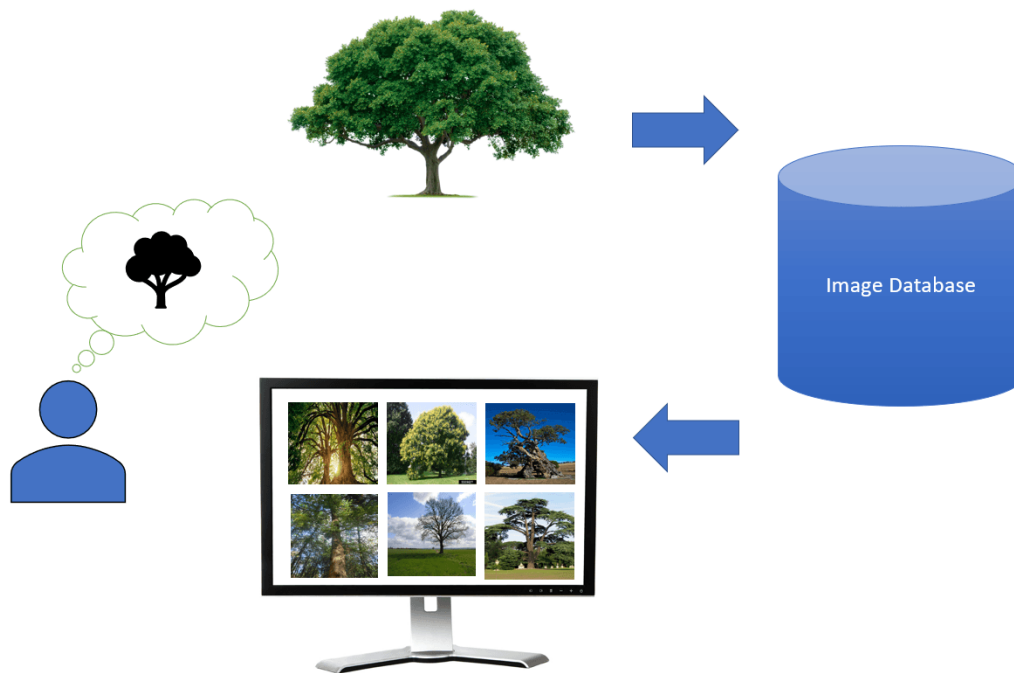


Figure 1-1: CBIR Implementation Diagram [2]

The difference between the low-level visual features that a computer can automatically extract from an image and the high-level concepts that humans perceive in that image is known as the semantic gap, and it is this gap that makes CBIR systems effective. To do this, CBIR systems use sophisticated algorithms and machine learning methods that examine an image's visual content and deduce its semantic and contextual meaning. By delivering more pertinent search results, this procedure not only improves user experience but also makes managing big image databases easier in situations where manual tagging and annotation are not feasible or sufficient. [3]

## **1.2. Examples of CBIR System Implementations**

A CBIR system can be implemented for an image to extract its features in many ways such as color histogram, color moments, and more, each one has its own image features which results in giving different outcomes and evaluations for each one:

### **1.2.1. Color Histograms**

Color histograms are a fundamental tool in CBIR analysis and interpretation. By counting the number of pixels that make up each color, these histograms depict the distribution of colors in an image. This technique offers a rapid and effective way to compare images based on their color composition and is especially resistant to changes in image size and orientation. The spatial distribution of colors within an image is not taken into consideration by color histograms, even though they are useful for locating images with comparable color patterns. [4]

### **1.2.2. Color Moments**

Color moments is Another important technique for extracting features in CBIR. Using this method, one can compute the statistical moments of an image's color distribution, including its mean, variance, and skewness. Color moments simplify the process of comparing and analyzing images by condensing the color information into a compact form. Compared to histograms, they offer a more thorough understanding of the color distribution and are especially helpful in situations where minute variations in color shades are significant. [5]

### **1.2.3. Other Ways of Implementations**

A CBIR system can be built in many other ways such local binary patterns (LBP) which is basically a texture operator that can be used for image classification, another way is the convolutional neural networks that extracts more complex and abstract features which better represents the semantic content of an image.

These various approaches allow CBIR systems to serve a broad spectrum of applications by providing advanced tools for retrieving images through thorough visual analysis.











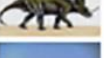

















## 2. System Implementation & Experimental Setup

### 2.1. The Used Dataset

The used dataset for this assignment is the waggle dataset that was taken from the Kaggle website in [This Link](#), this data set contains 1000 images, each 100 images representing a category, which makes the data set consists of 10 categories as shown in the table 2-1.

Table 2-1: Wang Dataset Categories [6]

Clusters No.	Semantic name	A sample of the images in each cluster					
1	African people village						
2	Beach						
3	Building						
4	Buses						
5	Dinosaurs						
6	Elephants						
7	Flowers						
8	Horses						
9	Mountains and glaciers						
10	Food						

### 2.2. The Used Libraries

The following libraries that can be used for the application:

- **PIL import Image:** this library was used to open images from the dataset.
- **Numpy:** Since this project is about getting features from images which are large many dimensional arrays, this library's mathematical features can be very useful in this assignment.
- **CV2:** This is the OpenCV library that allows the system to read images and extract their features in many ways.
- **Matplotlib:** This library is responsible for plotting images and graphs for the system.
- **Sklearn:** This library gives the functionality to calculate the area under a curve.
- **Scipy:** This library calculates all the needed statistics for all the given calculations.
- **Skimage:** This library is used to get the features of the local binary pattern system.

## 2.3. Image Loading Function

The images are loaded to be stored in two arrays with each image's index: the first array is contains the readings of all the images using the cv2 library, and the second array stores the basic image in order for it to be plotted later on, the function can be found in **Appendix 1**.

## 2.4. Choosing The Queries

The query images were chosen randomly, yet in the code they a static as the run for all the systems won't be done at once, for each category an image has been chosen as shown in figure 2-1.

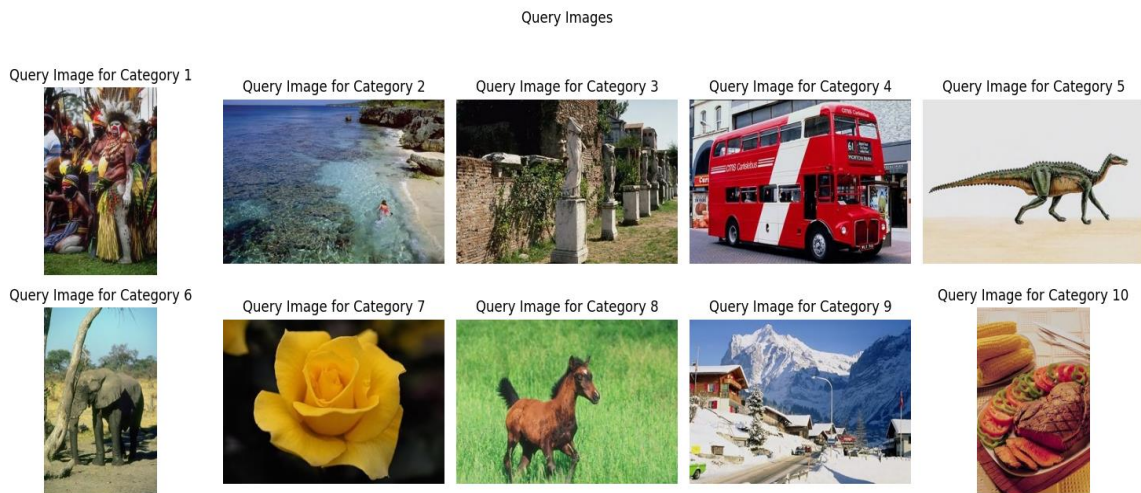


Figure 2-1: Query Images

## 2.5. Feature Extraction Functions

For this assignment there are three different implementations of the CBIR system which are:

### 2.5.1. Color Histogram

The color histogram feature extraction function takes the image and the desired number of bins and then extracts the features for the three channels (red, green, and blue) based on the number of bins then returns the features as a NumPy array, the code for this function can be found in **Appendix 2**.

### 2.5.2. Color Moments

The color moments feature extraction takes the image, weights values, and a Boolean value for extension. The image channel is taken using the OpenCV library to measure the following measurements as features:

- Mean
- Standard Deviation
- Skewness

These measurements are taken as features, they also can be weighed by the taken weights list. For the extended measurements which can be as extended features for the previous ones:

- Median
- Kurtosis
- Mode

The code for the color moments feature extraction can be found in **Appendix 3**.

### 2.5.3. Local Binary Patterns

This function takes the number of points which will be set to 24, the radius set to 8, and the eps which is set to  $e^{-7}$ . The image will be turned into a grey image and then the features are uniformly measured, then ranges between 2 and 3 to finally be normalized with the summation added with the eps. The code for this function can be found in **Appendix 4**.

## 2.6. Feature Distance Calculation

The distance between two sets of features is taken as a subtraction between the difference for each feature for both sets, then the differences are summed up as a distance value, the code for the function can be found in **Appendix 5**.

## 2.7. Retrieve Images for a Given Threshold

For a given threshold and a query image, the following steps are done:

1. Each distance will be measured between the query image and all the 1000 images.
2. The images with distance higher than the threshold are filtered out.
3. The images are sorted from the lower distance to the higher distance to be retrieved.
4. the first image is filtered out as it is the query image so the distance will be zero.

The code for this part can be found in **Appendix 6**.

## 2.8. System Evaluation

Given the retrieved images based on a threshold and the category of the given query image, the number retrieved relevant images (retrieved images from the same category) is calculated, so

that the true positive ratio, false positive ratio, precision, recall, and F1 score, all can be calculated as follows:

$$\text{true positive ratio} = \frac{\text{retrieved relevant}}{100 - 1}$$

$$\text{false positive ratio} = \frac{\text{retrived images} - \text{retrieved relevant}}{900}$$

$$\text{precision} = \frac{\text{retrieved relevant}}{\text{retrieved images}}$$

$$\text{recall} = \frac{\text{retrieved relevant}}{99}$$

$$\text{F1 score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

The function that evaluates the precision, recall, and F1 can be found in **Appendix 7**, while the code for the true positive and false positive ratio can be found in **Appendix 8**.

## 2.9. Finding Max Threshold

As the threshold sets the taken and not taken distances the values of the threshold will go on between zero and the maximum distance, the maximum needed threshold is the maximum distance value for all the measured distances in all the ten queries, the code for the this part can be found in **Appendix 9**.

## 2.10. Receiver Operating Characteristic Curve

The ROC curve presets the growth of the true positive values (as Y-axis) against the growth of the false positive vales (as X-axis) which the area under that curve can be taken as a performance measurement so in this assignment the curve is plotted and the area under it is calculated using the sklearn library function auc.

## 2.11. Plotting the Results

The matplotlib library is used to plot the top-5 similar images for a given 10 queries as the function for the code can be found in **Appendix 10**. The library is also used to plot the ROC curve for all the thresholds values of true positive rate and false positive rate as the code for it can be found in **Appendix 11**.

## 2.12. Implementations Values

As the CBIR system is implemented in different ways each way had its many implementations in different values too as following:

### 2.12.1. Color Histogram Implementation

For the color histogram the system was implemented on three different values of bins (120, 180, and 256) each time it measures the maximum threshold then goes on and evaluates the system on 100 different thresholds between zero and the maximum threshold finally getting the ROC curve values and plotting it, the code for the implementation using the given functions can be found in **Appendix 12**.

### 2.12.2. Color Moments Implementation

The color moments system was implemented many times, each time for different weights for its features (mean, standard deviation, and skewness) the first time with equal weights, then three times each time a feature has double the weight for the two other features, in the fifth run the extended features were added (median, kurtosis, and mode) as in this run the standard deviation and the median has half the other weights values table 2-2 shows the weights for each run.

Table 2-2: Weights Values

The Run	Weights		
1	33.3% mean	33.3% STD	33.3% skewness
2	50% mean	25% STD	25% skewness
3	25% mean	50% STD	25% skewness
4	25% mean	25% STD	50% skewness
5	16.7% mean, 8.3% STD 16.7% skewness, 8.3% median, 16.7% kurtosis, 16.7% mode		

At each run the maximum threshold then goes on and evaluates the system on 100 different thresholds between zero and the maximum threshold finally getting the ROC curve values and plotting it, the code for the implementation using the given functions can be found in **Appendix 13**.

### 2.12.2. Local Binary Patterns Implementation

The LBP is implemented with the following configurations (points = 24, radius=8 eps =  $e^{-7}$ ) the maximum threshold then goes on and evaluates the system on 100 different thresholds between zero and the maximum threshold finally getting the ROC curve values and plotting it, the code for the implementation using the given functions can be found in **Appendix 14**.



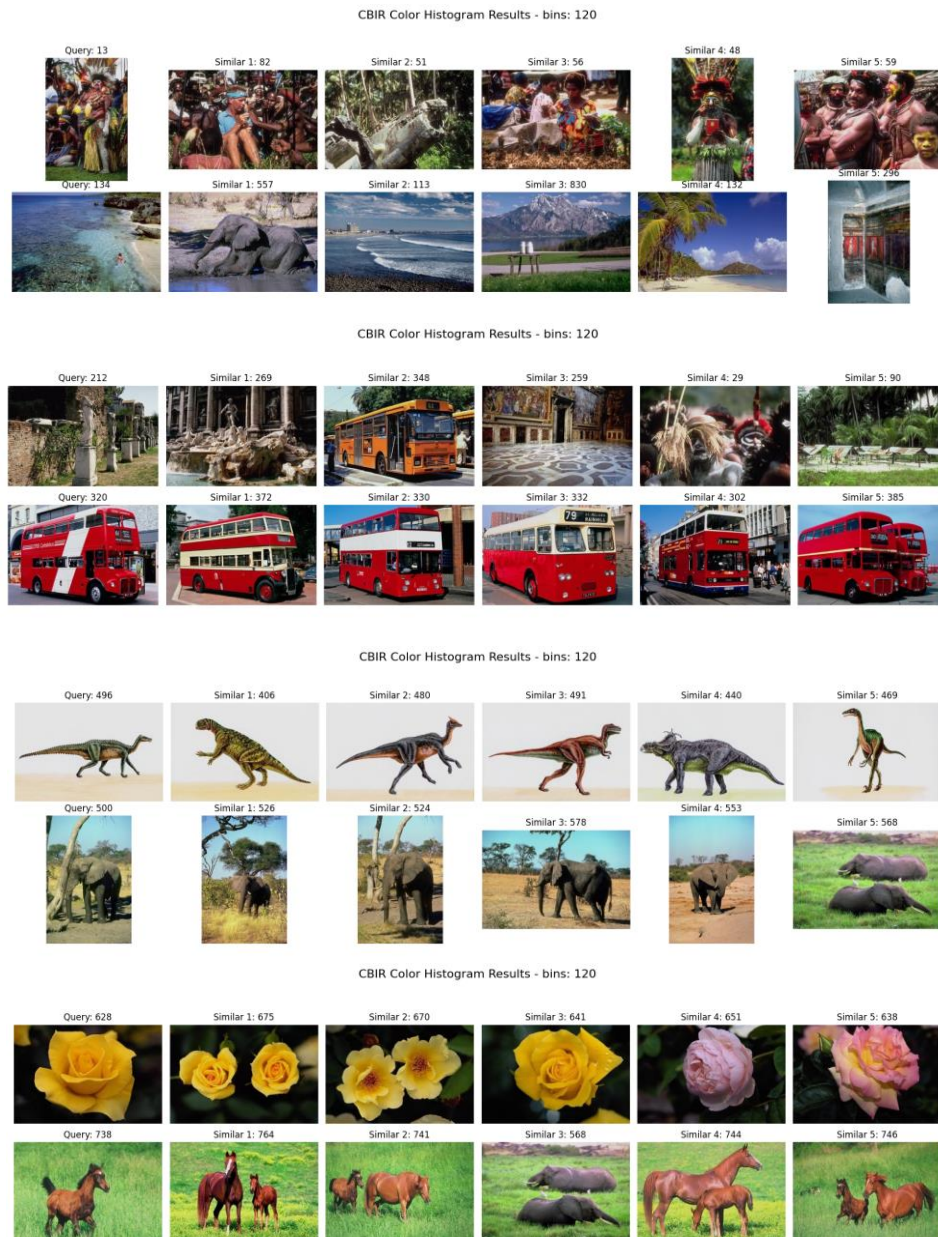
### 3. Results

This section presents the results for each system and its values where it shows the plot of the similar images, ROC curve, and evaluation metrics.

#### 3.1. Results for Color Histogram Implementation

##### 3.1.1. Color Histogram with 120 Bins

Figure 3-1 below shows the top-5 similar images for the 10 queries.





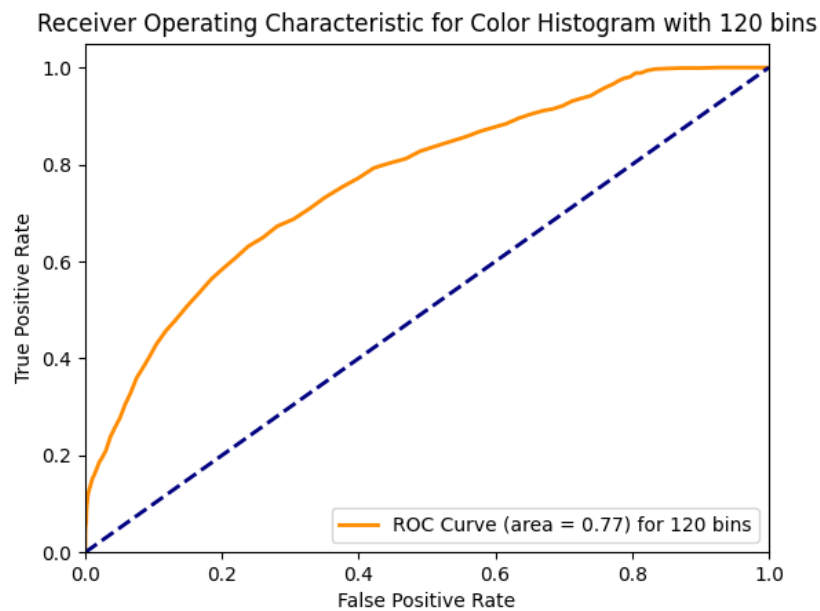
**Figure 3-1: Color Histogram 120 Bins Similar Images**

Table 3-1 below shows the metrics average values for 5 different threshold values:

**Table 3-1: Color Histogram 120 Bins Retrieved Metrics**

Threshold	Precision	Recall	F1	Time	Retrieved Images
0.25	0.0	0.0	0.0	2ms	0.0
0.74	0.65	0.1	0.14	2ms	12.5
1.23	0.43	0.46	0.38	2ms	151.2
1.73	0.22	0.85	0.3	2ms	567.0
2.22	0.15	0.98	0.24	2ms	829.4

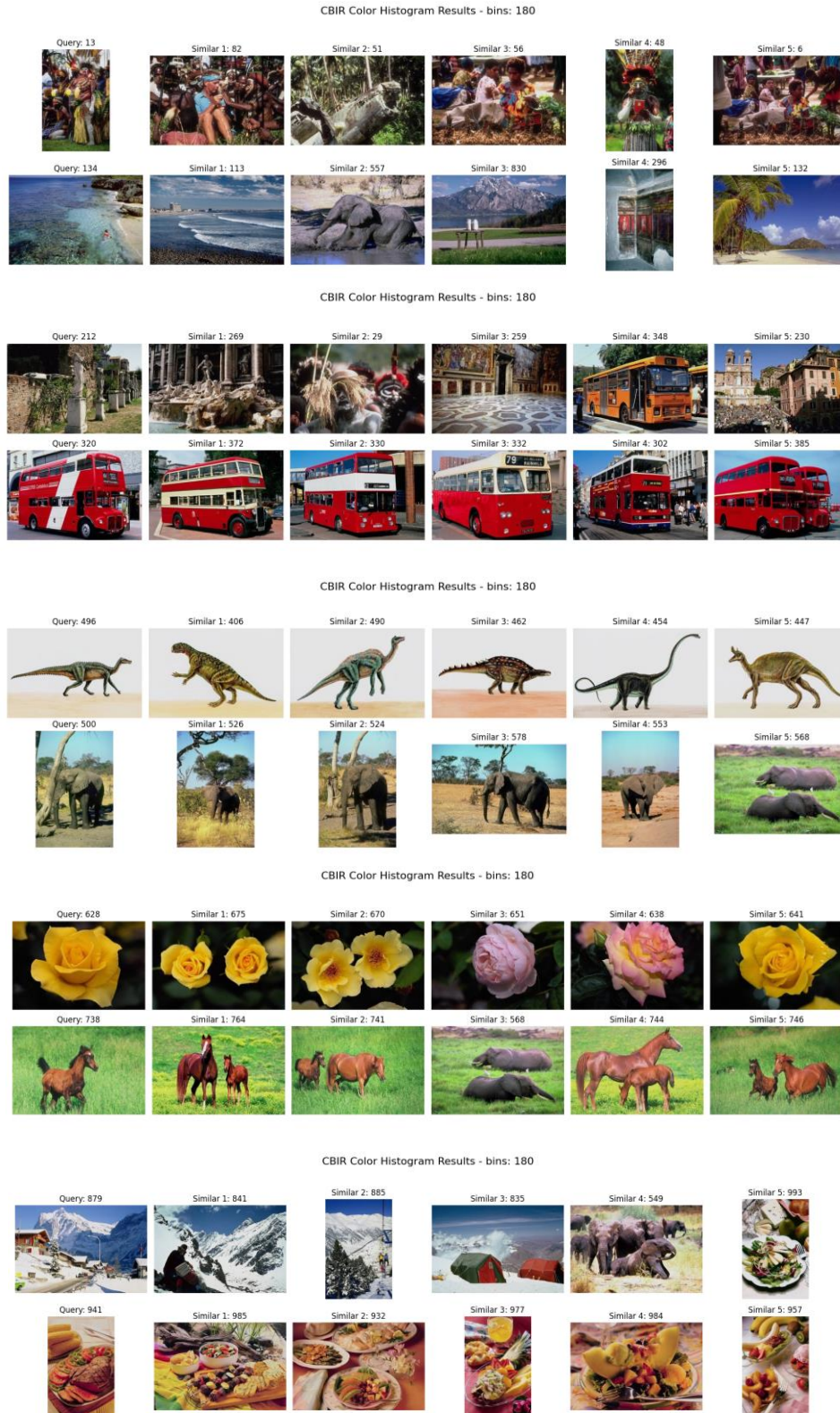
Figure 3-2 shows the ROC curve for the color histogram with 120 pins with the area under it = 0.77.



**Figure 3-2: ROC Curve for 120 Pins Color Histogram**

### 3.1.2. Color Histogram with 180 Bins

Figure 3-3 below shows the top-5 similar images for the 10 queries.



**Figure 3-3: Color Histogram 120 Bins Similar Images**

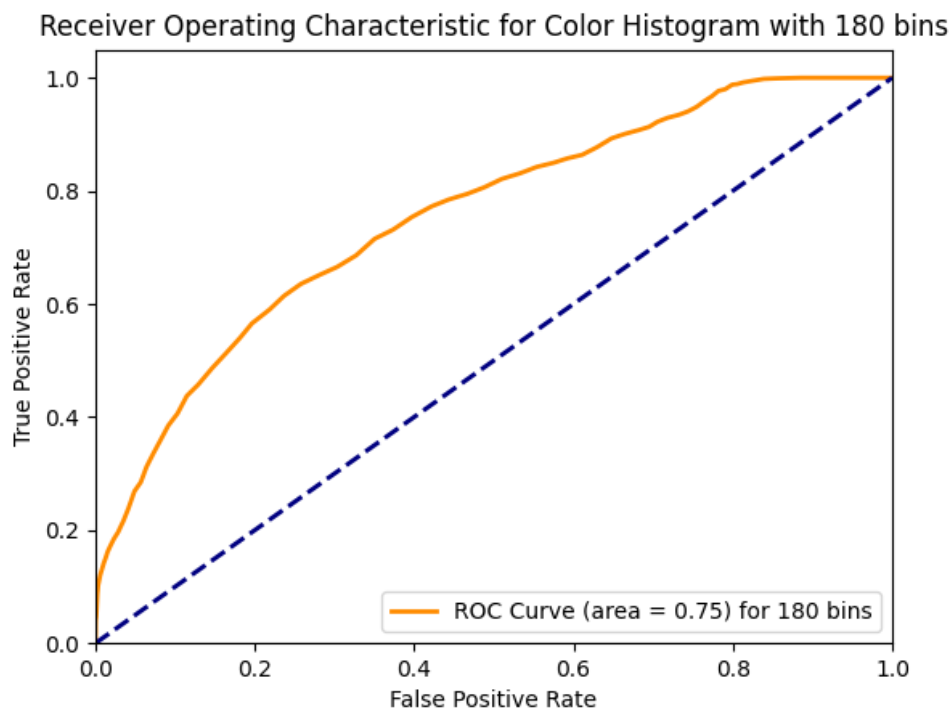
Table 3-2 below shows the metrics average values for 5 different threshold values:



**Table 3-2: Color Histogram 180 Bins Retrieved Metrics**

Threshold	Precision	Recall	F1	Time	Retrieved Images
0.25	0.0	0.0	0.0	2ms	0.0
0.74	0.65	0.09	0.13	2ms	11.8
1.23	0.43	0.43	0.37	2ms	146.5
1.73	0.22	0.83	0.29	2ms	562.4
2.22	0.16	0.98	0.25	3ms	822.8

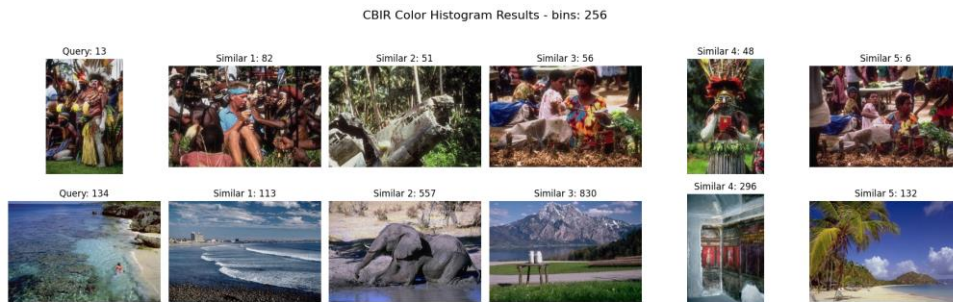
Figure 3-4 shows the ROC curve for the color histogram with 180 pins with the area under it = 0.75.



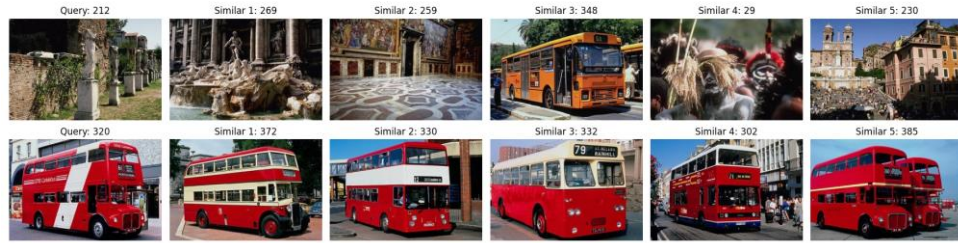
**Figure 3-4: ROC Curve for 180 Pins Color Histogram**

### 3.1.3. Color Histogram with 256 Bins

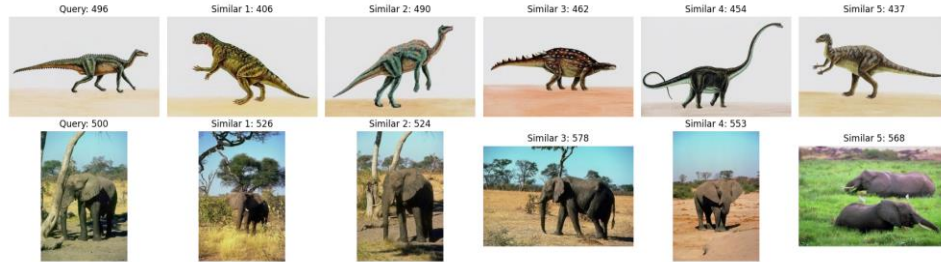
Figure 3-5 below shows the top-5 similar images for the 10 queries.



CBIR Color Histogram Results - bins: 256



CBIR Color Histogram Results - bins: 256



CBIR Color Histogram Results - bins: 256



CBIR Color Histogram Results - bins: 256



Figure 3-5: Color Histogram 256 Bins Similar Images

Table 3-3 below shows the metrics average values for 5 different threshold values:

Table 3-3: Color Histogram 256 Bins Retrieved Metrics

Threshold	Precision	Recall	F1	Time	Retrieved Images
0.25	0.0	0.0	0.0	2ms	0.0
0.74	0.68	0.09	0.12	2ms	10.7
1.23	0.44	0.42	0.36	3ms	139.9
1.73	0.22	0.83	0.29	3ms	557.6
2.22	0.16	0.98	0.25	3ms	821.9

Figure 3-6 shows the ROC curve for the color histogram with 256 pins with the area under it = 0.75.

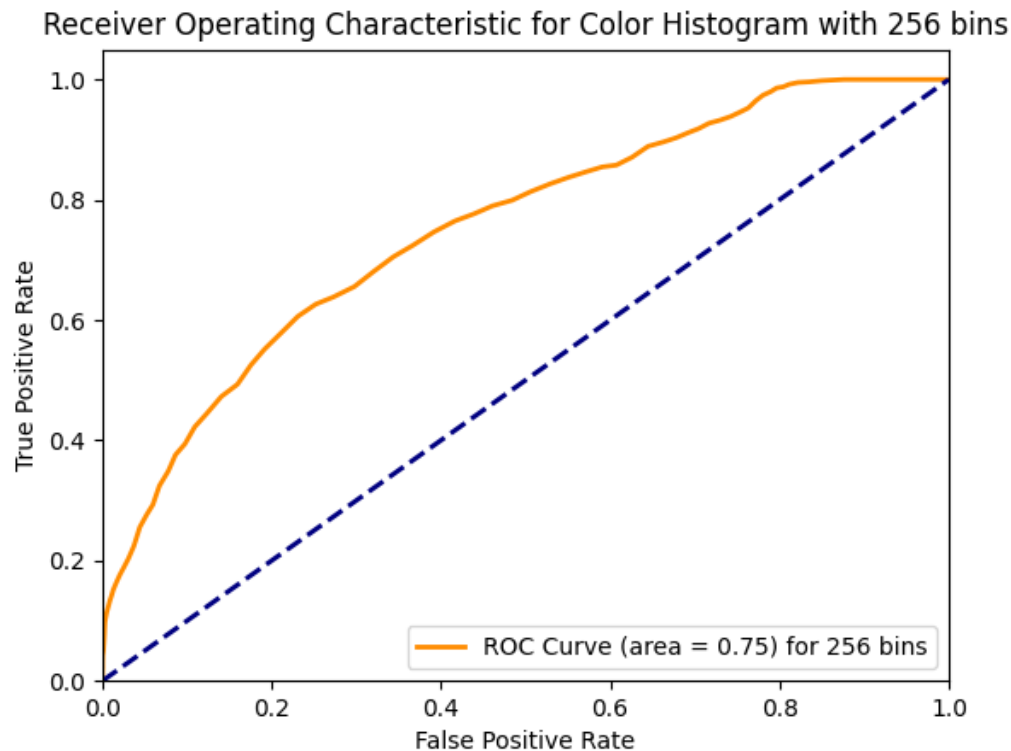


Figure 3-6: ROC Curve for 256 Pins Color Histogram

#### 3.1.4. Color Histogram Discussion

As shown in the past three results. When the number of taken bins increases the system performance is reduced as the 120 bins gave the best results, yet when the number of bins was increased to 180 the average precision, recall, F1 score, and the ROC AUC were decreased, the performance decreased more when the number of bins was increased to 256 yet the performance difference between the 180 bins and the 256 is almost the same.

Note that the precession decreases linearly with the threshold increases while the recall linearly increases, the best threshold value that gave the best F1 score lays near the middle between the zero and the max distance.

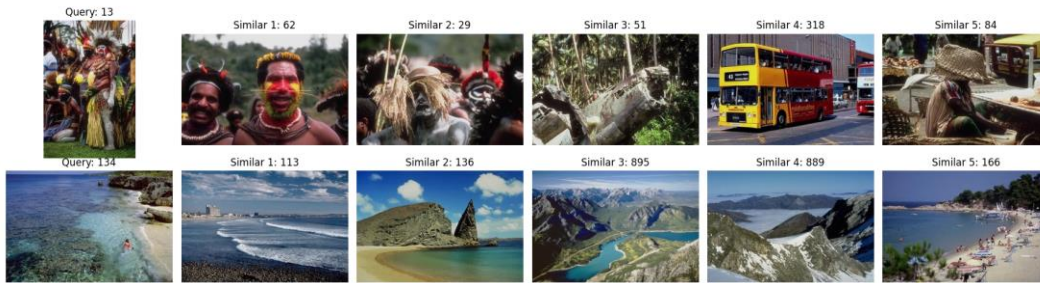


## 3.2. Results from Color Moments Implementation

### 3.2.1. Color Moments Equal Weights

For this part the weights for the mean STD and skewness are all equally the same. Figure 3-7 below shows the top-5 similar images for the 10 queries.

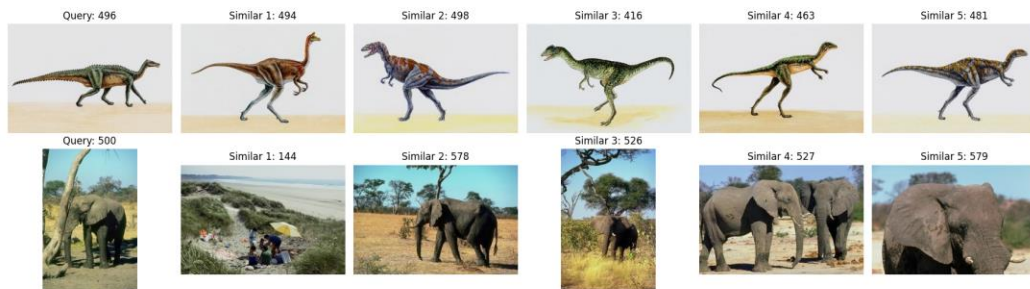
CBIR Color Moments Results - weights: 1



CBIR Color Moments Results - weights: 1

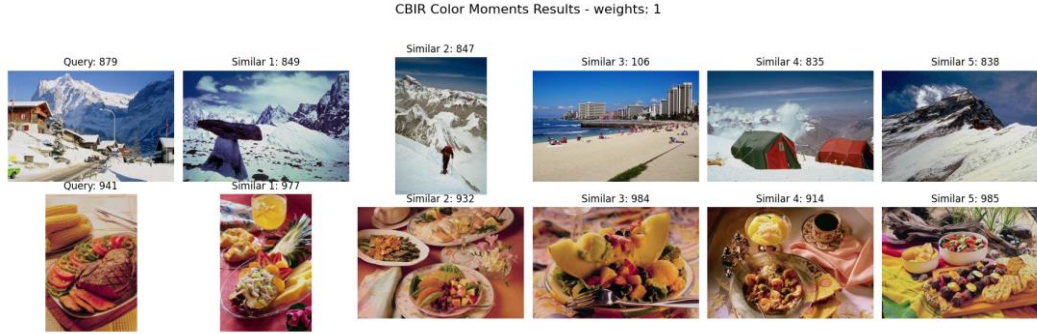


CBIR Color Moments Results - weights: 1



CBIR Color Moments Results - weights: 1





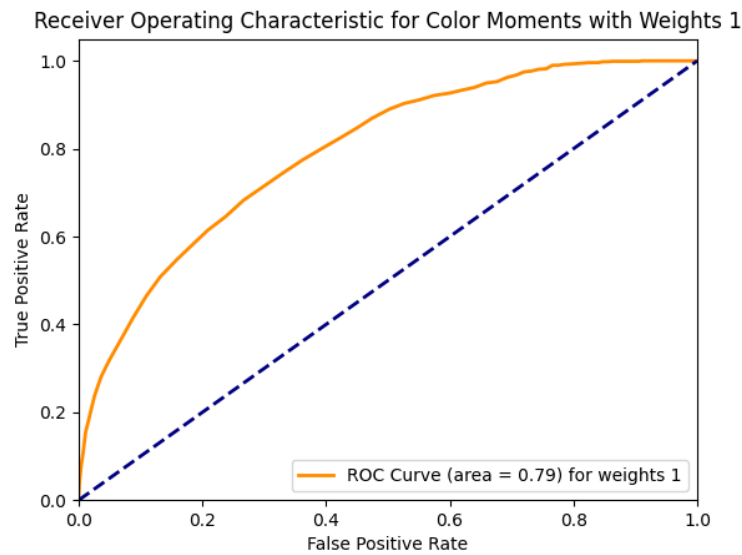
**Figure 3-7: Color Moments Equal Weights Similar Images**

Table 3-4 below shows the metrics average values for 5 different threshold values:

**Table 3-4: Color Moments Equal Weights Retrieved Metrics**

Threshold	Precision	Recall	F1	Time	Retrieved Images
30.97	0.65	0.28	0.33	2ms	60.3
92.91	0.22	0.91	0.32	2ms	585.7
154.85	0.14	0.99	0.23	2ms	847.4
216.8	0.1	1.0	0.19	2ms	959.5
278.74	0.1	1.0	0.18	2ms	996.1

Figure 3- shows the ROC curve for the color moments for all equal weights with the area under it = 0.79.



**Figure 3-8: ROC Curve for Color Moments Equal Weights**

### 3.2.2. Color Moments Mean Weight Double STD & Skewness

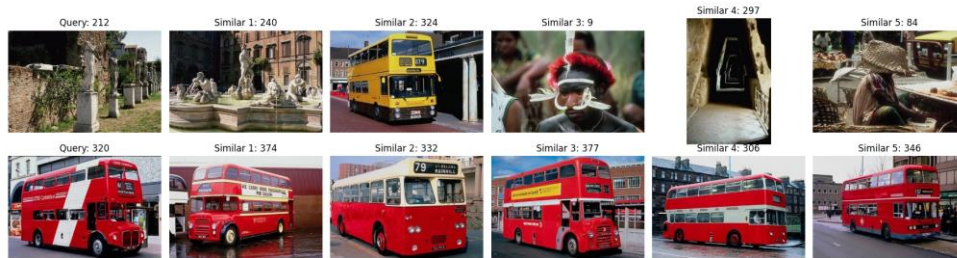
For this part the weight for the mean is double the weight of the STD and the skewness. Figure 3-9 below shows the top-5 similar images for the 10 queries.



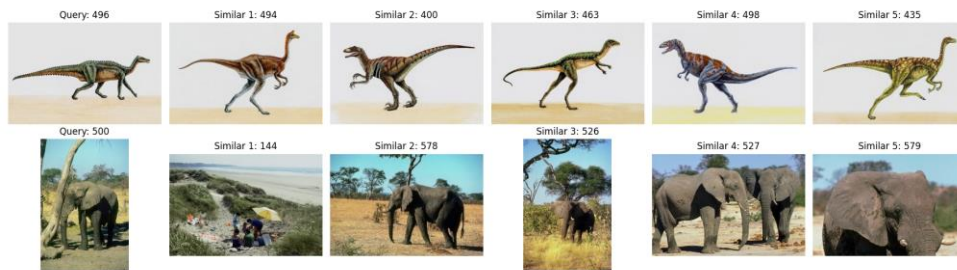
CBIR Color Moments Results - weights: 2



CBIR Color Moments Results - weights: 2



CBIR Color Moments Results - weights: 2



CBIR Color Moments Results - weights: 2



CBIR Color Moments Results - weights: 2

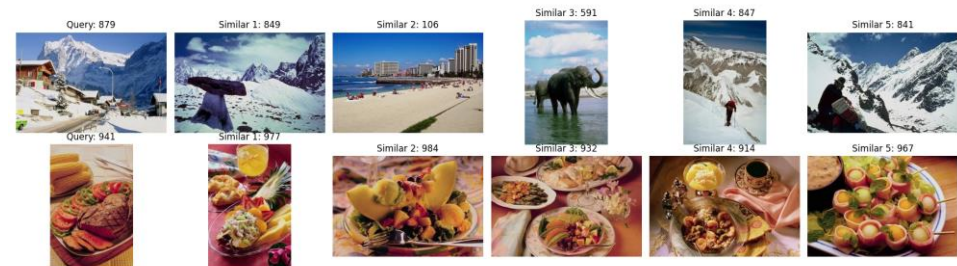


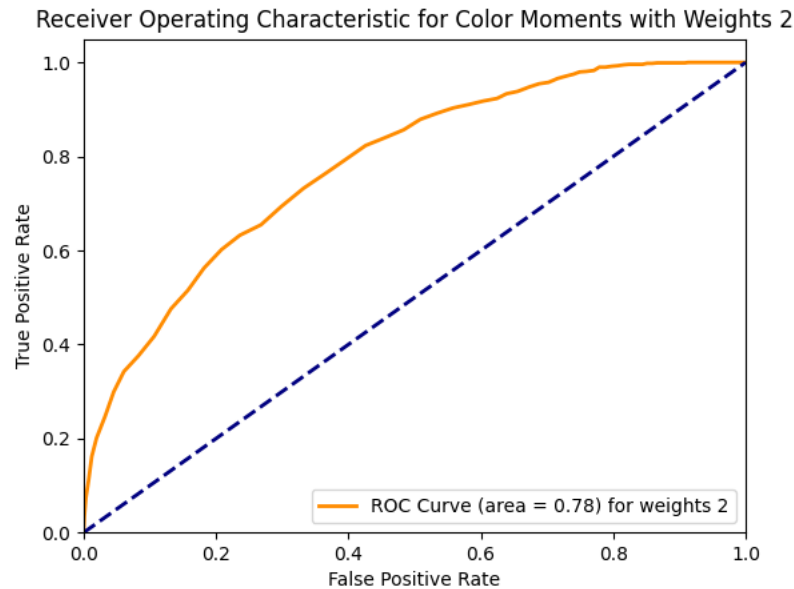
Figure 3-9: Color Moments Double Mean Weights Similar Images

Table 3-5 below shows the metrics average values for 5 different threshold values:

**Table 3-5: Color Moments Double Mean Weights Retrieved Metrics**

Threshold	Precision	Recall	F1	Time	Retrieved Images
15.4	0.54	0.34	0.35	0.2ms	88.5
46.18	0.2	0.92	0.3	0.2ms	635.2
76.98	0.13	0.99	0.23	0.2ms	852.6
107.77	0.1	1.0	0.19	0.2ms	959.8
138.56	0.1	1.0	0.18	0.2ms	996.1

Figure 3-10 shows the ROC curve for the color moments for double weights for the mean with the area under it = 0.78.

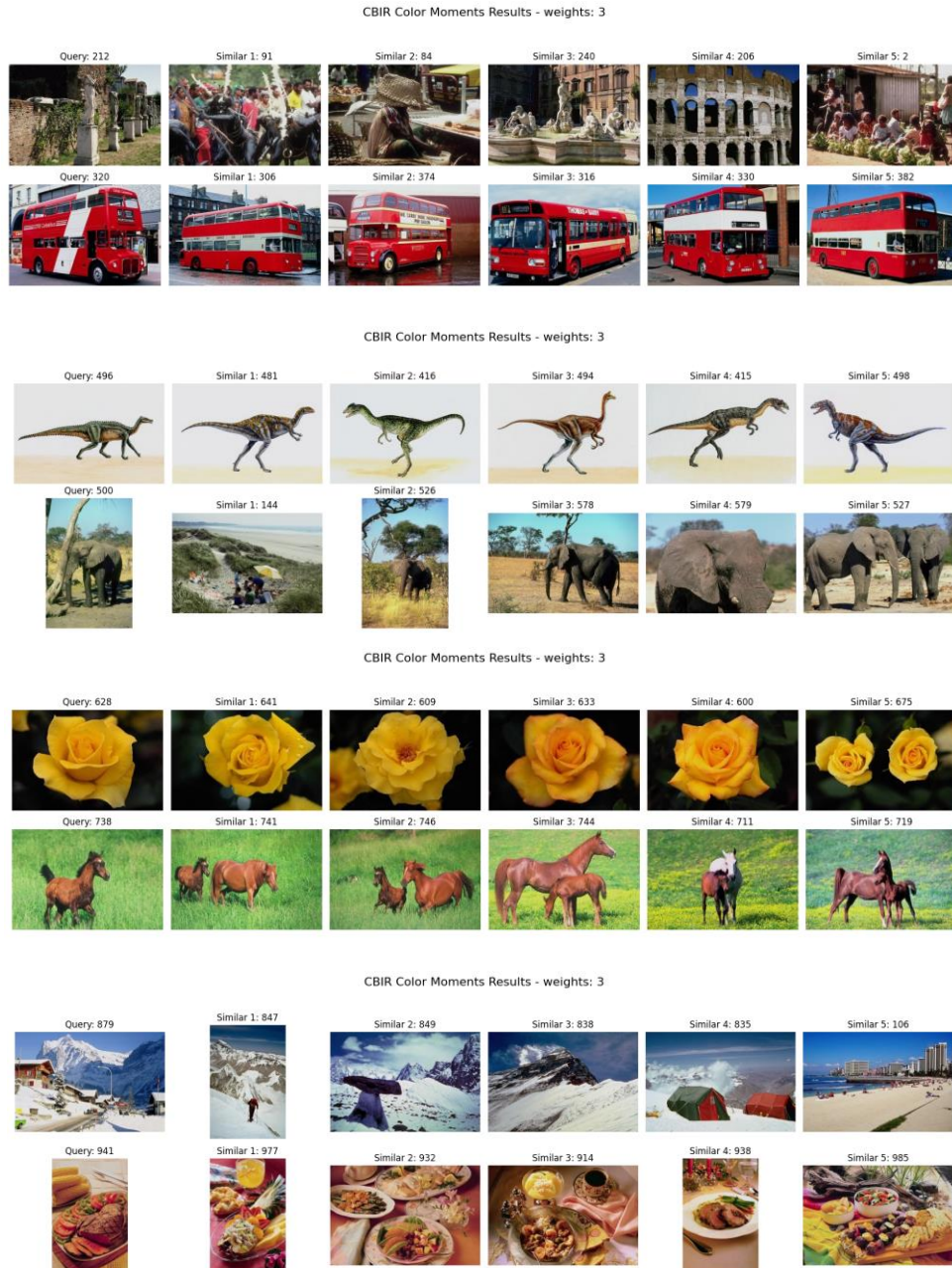


**Figure 3-10: ROC Curve for Color Moments Double Mean Weights**

### 3.2.3. Color Moments STD Weight Double Mean & Skewness

For this part the weight for the STD is double the weight of the mean and the skewness. Figure 3-11 below shows the top-5 similar images for the 10 queries.





**Figure 3-11: Color Moments Double STD Weights Similar Images**

Table 3-6 below shows the metrics average values for 5 different threshold values:

**Table 3-6: Color Moments Double STD Weights Retrieved Metrics**

Threshold	Precision	Recall	F1	Time	Retrieved Images
7.96	0.74	0.17	0.23	2ms	31.1
23.87	0.28	0.82	0.36	2ms	445.3
39.79	0.14	0.99	0.24	2ms	798.0
55.7	0.1	1.0	0.19	2ms	957.7
71.62	0.1	1.0	0.18	2ms	994.5



Figure 3-12 shows the ROC curve for the color moments for double weights for the STD with the area under it = 0.8.

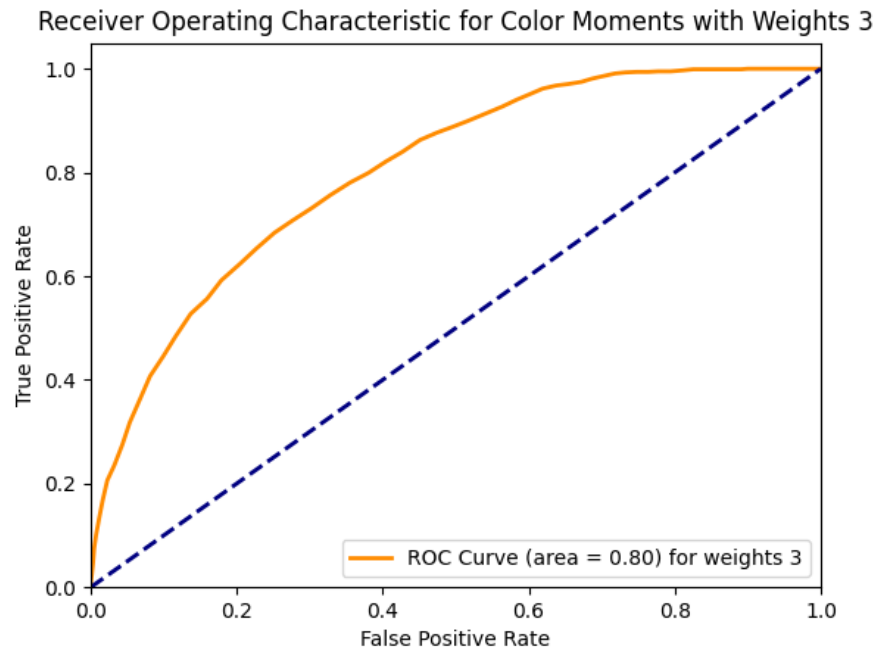
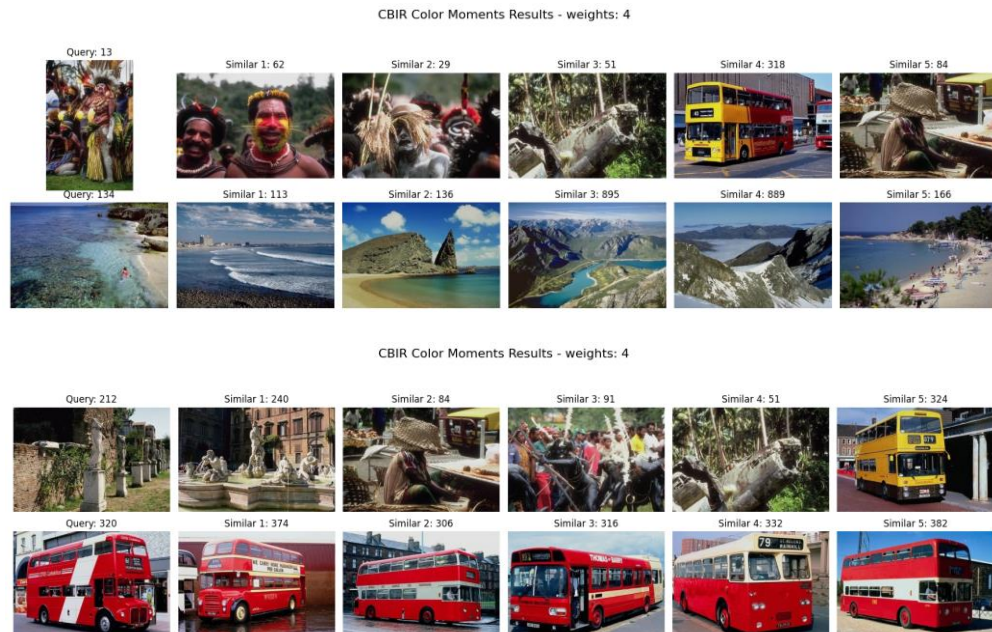
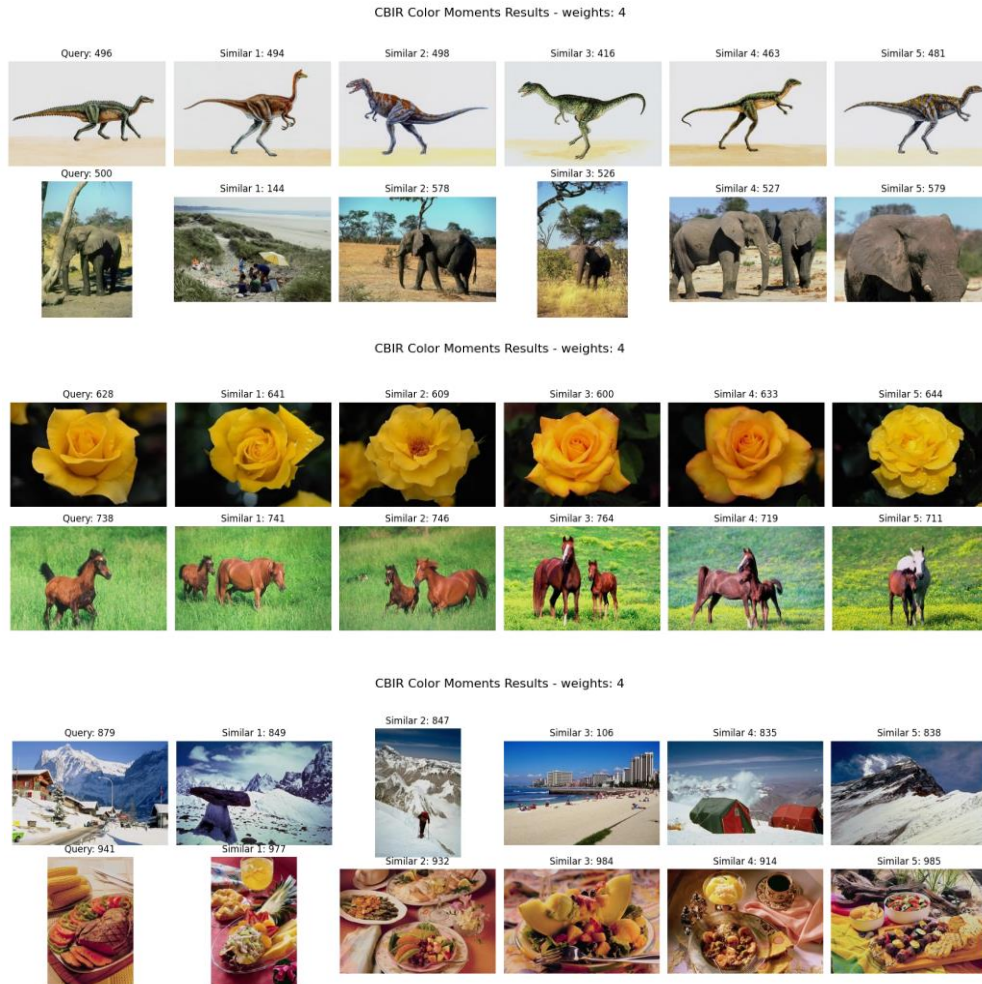


Figure 3-12: ROC Curve for Color Moments Double STD Weights

### 3.2.4. Color Moments Skewness Weight Double Mean & STD

For this part the weight for the STD is double the weight of the mean and the skewness. Figure 3-13 below shows the top-5 similar images for the 10 queries.





**Figure 3-13: Color Moments Double Skewness Weights Similar Images**

Table 3-7 below shows the metrics average values for 5 different threshold values:

**Table 3-7: Color Moments Double Skewness Weights Retrieved Metrics**

Threshold	Precision	Recall	F1	Time	Retrieved Images
7.75	0.65	0.28	0.32	2ms	60.4
23.26	0.22	0.91	0.32	2ms	586.1
38.77	0.14	1.0	0.23	2ms	847.9
54.27	0.1	1.0	0.19	2ms	959.6
69.78	0.1	1.0	0.18	2ms	996.1

Figure 3-14 shows the ROC curve for the color moments for double weights for the Skewness with the area under it = 0.79.

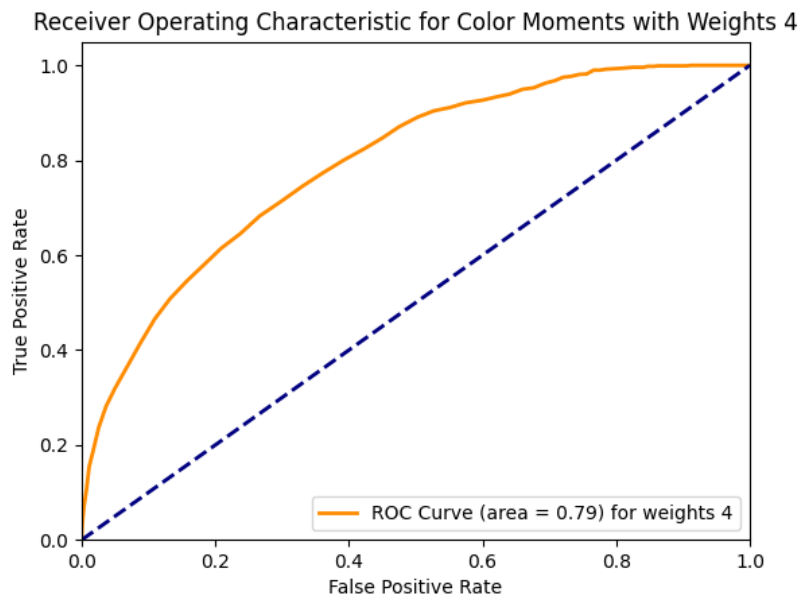
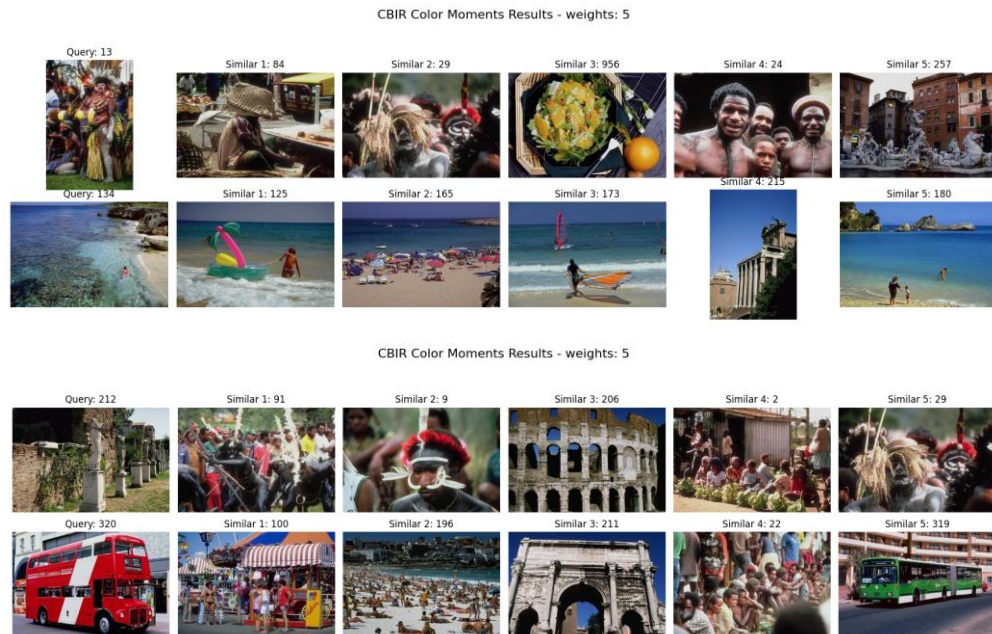


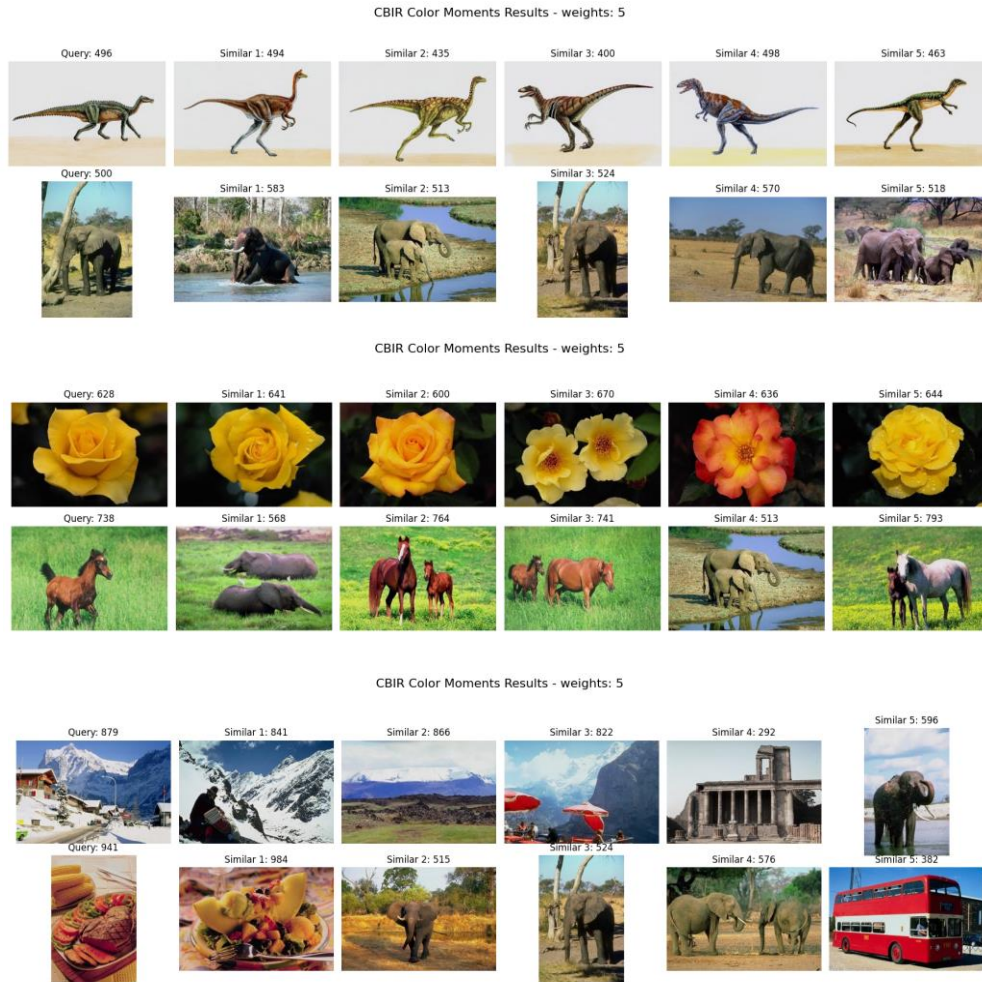
Figure 3-14: ROC Curve for Color Moments Double Skewness Weights

### 3.2.5. Color Moments with Extended Features

For this part the in addition for the mean STD and the skewness the median, Mode, and Kurtosis are added. Figure 3-15 below shows the top-5 similar images for the 10 queries.







**Figure 3-15: Color Moments Extended Features Similar Images**

Table 3-8 below shows the metrics average values for 5 different threshold values:

**Table 3-8: Color Moments Double Extended Features Retrieved Metrics**

Threshold	Precision	Recall	F1	Time	Retrieved Images
10.68	0.48	0.22	0.24	2ms	46.3
32.05	0.2	0.64	0.29	2ms	395.1
53.42	0.13	0.88	0.22	2ms	733.1
74.79	0.11	0.98	0.2	2ms	892.5
96.16	0.1	1.0	0.18	2ms	983.8

Figure 3-16 shows the ROC curve for the color moments with extended features (median, mode, kurtosis) with the area under it = 0.71.

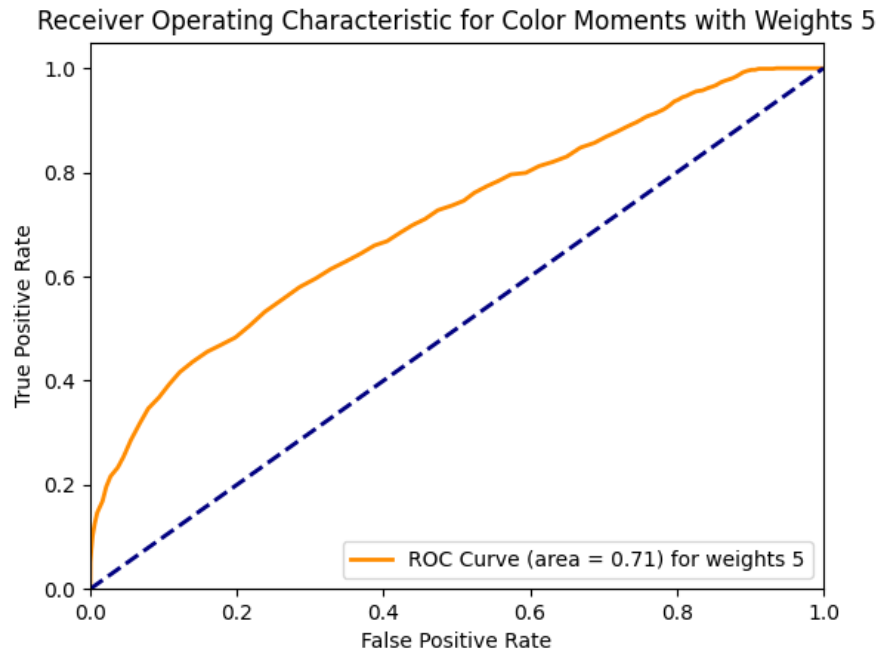


Figure 3-16: ROC Curve for Color Moments Extended Features

### 3.2.6. Color Moments Discussion

Based on the previous results it was concluded that the color moments give better results than the histogram implementation, as the extracted features are more accurate, it was also concluded that the STD and the skewness weight increase gave a little enhancement to the system.

Adding the extended features (median, Mode, and Kurtosis) have made the ROC AUC curve way worse, although it gave a slightly better results for the precision, and F1 score, as the bigger weight shall be given to the original features which wasn't added in this part causing the downfall in the ROC AUC.

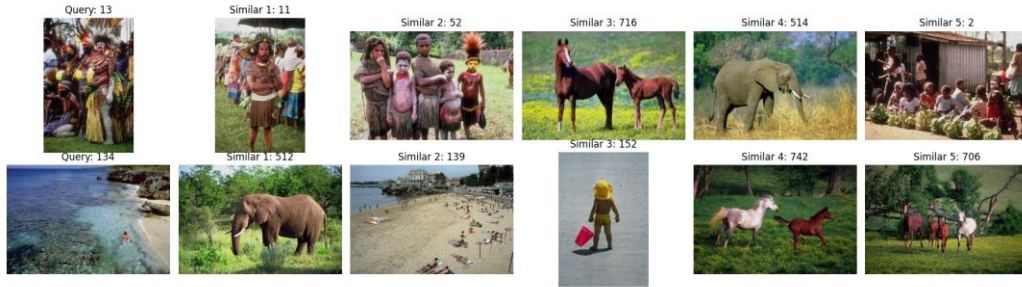
Unlike the color histogram implementation, the color moments threshold value cause logarithmic increase in the recall and logarithmic decrease in the precision what was saw for all different weights.

### 3.3. Results from Local Binary Patterns

#### 3.3.1. LBP With 24 Points

For this part the local binary patterns system is implemented with the following configurations (points = 24, radius=8 eps =  $e^{-7}$ ). Figure 3-17 below shows the top-5 similar images for the 10 queries.

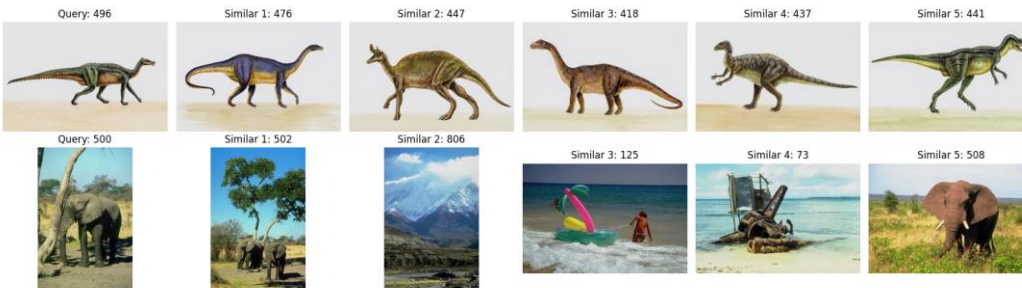
CBIR Results for LBP



CBIR Results for LBP

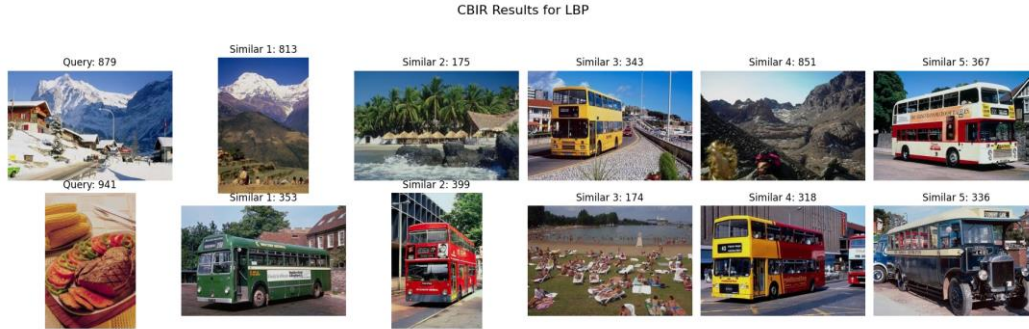


CBIR Results for LBP



CBIR Results for LBP





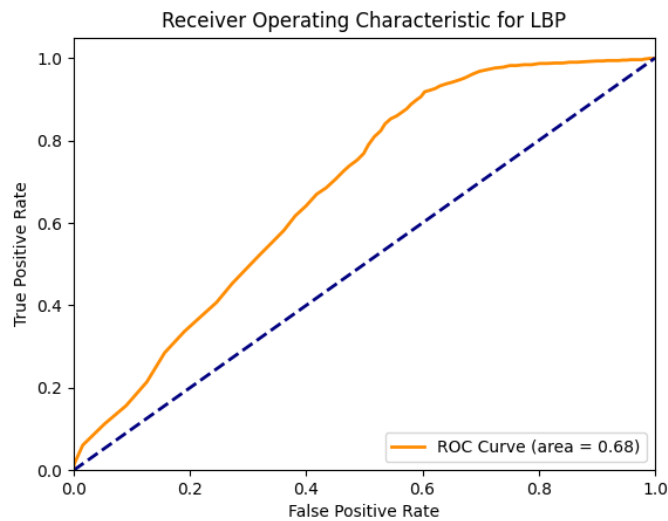
**Figure 3-17: LBP Extended Features Similar Images**

Table 3-9 below shows the metrics average values for 5 different threshold values:

**Table 3-9: LBP Features Retrieved Metrics**

Threshold	Precision	Recall	F1	Time	Retrieved Images
0.12	0.28	0.7	0.31	2ms	474.5
0.25	0.2	0.94	0.3	2ms	669.4
0.38	0.11	0.99	0.2	2ms	898.0
0.5	0.1	0.99	0.18	2ms	992.7
0.63	0.1	1.0	0.18	2ms	999

Figure 3-18 shows the ROC curve for the color moments with extended features (median, mode, kurtosis) with the area under it = 0.71.



**Figure 3-18: ROC Curve for LBP Extended Features**

### 3.3.2. LBP Discussion

The local binary patterns result such as the color moments are logarithmic for the thresholds, yet its results are much worse, that could possibly be a result of bad configurations setup as shown in the retrieved number of images and the ROC curve too.



## 4. Conclusion

In conclusion this assignment shows the importance and use of the content-based image retrieval system, as it can filter and give images based on the content inside them, which can be used in many applications today in searching for items to buy, discovering locations, finding similarities and more.

This assignment gives a better understanding of how features can be extracted from images to build AI models that can find search to find similarities with other images or retrieving data, it also shows how content-based systems can be evaluated as how their performance and accuracy can be measured across with threshold setup and mistake avoidance in order to know if weather they shall be implemented or not for actual work.

It was noticed that small topic images (such as the 5<sup>th</sup> and 7<sup>th</sup> category) can give better results than complex many topic images (first, second and 9<sup>th</sup> categories) as they can include different frequencies of colors, so the topic can't really be measured by the system as the easier ones.

Systems like the CBIR need to have more than a dependency such as color, shapes, and patterns as one feature is not usually enough resulting in focusing on retrieving images with no relation but that one feature which was experimented on in this assignment.



## 5. References

- [1]. Wikimedia Foundation. (2023, December 2). *Content-based image retrieval*. Wikipedia.  
[https://en.wikipedia.org/wiki/Content-based\\_image\\_retrieval](https://en.wikipedia.org/wiki/Content-based_image_retrieval) Accessed on January 7th, 2024.
- [2]. baeldung, W. by: (2022, December 10). *What is content-based image retrieval?* Baeldung on Computer Science. <https://www.baeldung.com/cs/cbir-tbir> Accessed on January 7th, 2024.
- [3]. Latif, A. (2019, August 26). *Content-based image retrieval and feature extraction: A comprehensive review*. Mathematical Problems in Engineering.  
<https://www.hindawi.com/journals/mpe/2019/9658350/> Accessed on January 7th, 2024.
- [4]. Content-based image retrieval using color moment and Gabor texture feature. (n.d.).  
[https://www.researchgate.net/publication/284543109\\_Content-Based\\_Image\\_Retrieval\\_using\\_Color\\_Moment\\_and\\_Gabor\\_Texture\\_Feature](https://www.researchgate.net/publication/284543109_Content-Based_Image_Retrieval_using_Color_Moment_and_Gabor_Texture_Feature)  
Accessed on January 7th, 2024.
- [5]. Content-based image retrieval using color moment and Gabor texture feature. (n.d.-a).  
[https://www.researchgate.net/publication/284543109\\_Content-Based\\_Image\\_Retrieval\\_using\\_Color\\_Moment\\_and\\_Gabor\\_Texture\\_Feature](https://www.researchgate.net/publication/284543109_Content-Based_Image_Retrieval_using_Color_Moment_and_Gabor_Texture_Feature)  
Accessed on January 7th, 2024.
- [6]. Sample wang dataset images in 10 semantic classes - researchgate. (n.d.-c).  
[https://www.researchgate.net/figure/Sample-WANG-dataset-images-in-10-semantic-classes\\_fig3\\_351514630](https://www.researchgate.net/figure/Sample-WANG-dataset-images-in-10-semantic-classes_fig3_351514630) Accessed on January 7th, 2024.

## 6. Appendix

### 6.1. Appendix 1

```
def load_images_from_folder(folder):
    images_CV = []
    images_plot = []
    for i in range(1000):
        filename = os.path.join(folder, f"{i}.jpg")
        try:
            img = cv2.imread(filename)
            images_CV.append(img)
            img = Image.open(filename)
            images_plot.append(img)
        except IOError:
            print(f"Error opening {filename}")
            continue
    return images_CV, images_plot
```

### 6.2. Appendix 2

```
def extract_color_histogram(image, bins):
    colors = cv2.split(image)
    histograms = [cv2.calcHist([color], [0], None, [bins], [0, 256]) for color in colors]
    histograms = [cv2.normalize(histogram, histogram).flatten() for histogram in histograms]
    return np.array(histograms)
```

### 6.3. Appendix 3

```
def extract_color_moments(image_cv, weights, extended):
    moments = []
    for i in range(3):
        channel = image_cv[:, :, i]
        mean = np.mean(channel)
        std = np.std(channel)
        skewness = np.mean((channel - mean)**3) / std**3
        if extended:
            kurtosis = stats.kurtosis(channel, axis=None)
            median = np.median(channel)
            mode = stats.mode(channel).mode[0] if stats.mode(channel).count[0] > 1 else 0
            moments.extend([mean*0.2, std*0.1, skewness*0.2, kurtosis*0.2, median*0.1, mode*0.2])
        else:
            moments.extend([mean * weights[0], std * weights[1], skewness * weights[2]])
    return np.array(moments)
```

### 6.4. Appendix 4

```
def extract_lbp_features(image, numPoints=24, radius=8, eps=1e-7):
```

```

if len(image.shape) == 3:
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
lbp = feature.local_binary_pattern(image, numPoints, radius, method="uniform")
(hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, numPoints + 3), range=(0, numPoints + 2))
hist = hist.astype("float")
hist /= (hist.sum() + eps)
return hist

```

## 6.5. Appendix 5

```

def calculate_distance(features1, features2):
    return np.linalg.norm(features1 - features2)

```

## 6.6. Appendix 6

```

def retrieve_images(query_hist, all_histograms, threshold):
    distances = [(index, calculate_distance(query_hist, hist)) for index, hist in enumerate(all_histograms)]
    sorted_distances = sorted((index_distance for index_distance in distances), key=lambda x: x[1])[1:]
    filtered_distances = [element for element in sorted_distances if element[1] <= threshold]
    return [index for index, distance in filtered_distances]

```

## 6.7. Appendix 7

```

def compute_metrics(retrieved_indices, query_label):
    relevant_indices = set(range(query_label*100, (query_label+1)*100))
    retrieved_relevant = relevant_indices.intersection(set(retrieved_indices))
    precision = len(retrieved_relevant) / len(retrieved_indices) if len(retrieved_indices) > 0 else 0
    recall = len(retrieved_relevant) / 99
    f1_score = 2 * (precision * recall) / (precision + recall) if (precision + recall) != 0 else 0
    return precision, recall, f1_score

```

## 6.8. Appendix 8

```

def calculate_tpr_fpr(query_index, retrieved_indices):
    query_label = query_index // 100
    true_positives = len([i for i in retrieved_indices if i // 100 == query_label])
    false_positives = len(retrieved_indices) - true_positives
    tpr = true_positives / 99
    fpr = false_positives / 900
    return tpr, fpr

```

## 6.9. Appendix 9

```

def find_max_threshold(query_images, histograms):
    max_threshold = max([max([calculate_distance(histogram, histograms[query_image])
                             for histogram in histograms]) for query_image in query_images])
    return max_threshold

```

## 6.10. Appendix 10

```
def plot_all_queries(results, dataset, bins):
    num_columns = 6
    num_rows = 2
    for i in range(0, len(results), num_rows):
        fig, axs = plt.subplots(num_rows, num_columns, figsize=(3 * num_columns, 3 * num_rows))

        for row in range(num_rows):
            if i + row < len(results):
                result = results[i + row]
                query_index = result['query_index']
                retrieved_indices = result['retrieved']
                query_img = dataset[query_index]
                axs[row, 0].imshow(query_img)
                axs[row, 0].set_title(f"Query: {query_index}")
                axs[row, 0].axis('off')
                max_similar = min(len(retrieved_indices), num_columns - 1)
                for j in range(1, num_columns):
                    if j <= max_similar:
                        idx = retrieved_indices[j - 1]
                        similar_img = dataset[idx]
                        axs[row, j].imshow(similar_img)
                        axs[row, j].set_title(f"Similar {j}: {idx}")
                        axs[row, j].axis('off')
                    else:
                        axs[row, j].axis('off')
            else:
                for col in range(num_columns):
                    axs[row, col].axis('off')

        plt.suptitle(f"CBIR Color Histogram Results - bins: {bins}", fontsize=16)
        plt.tight_layout(rect=[0, 0.03, 1, 0.95])
        plt.show()
```

## 6.11. Appendix 11

```
def plot_ROC(roc_auc, bin, fprs, tprs):
    plt.figure()
    plt.plot(fprs, tprs, color='darkorange', lw=2, label=f"ROC Curve (area = {roc_auc:.2f}) for {bin} bins")
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f"Receiver Operating Characteristic for Color Histogram with {bin} bins")
    plt.legend(loc="lower right")
    plt.show()
```

## 6.12. Appendix 12

```

folder = "dataset"
dataset_CV, dataset_plot = load_images_from_folder(folder)
query_indexes = [13, 134, 212, 320, 496, 500, 628, 738, 879, 941]
bins = [120, 180, 256]

for bin in bins:
    tprs = []
    fprs = []
    all_histograms = [extract_color_histogram(img, bin) for img in dataset_CV]
    max_threshold = find_max_threshold(query_indexes, all_histograms)
    thresholds = np.linspace(0, max_threshold, 100)
    for i, threshold in enumerate(thresholds):
        tpr_avg = 0
        fpr_avg = 0
        results = []
        for query_label, query_index in enumerate(query_indexes):
            query_hist = all_histograms[query_index]
            start_time = time.time()
            retrieved_indices = retrieve_images(query_hist, all_histograms, threshold)
            precision, recall, f1_score = compute_metrics(retrieved_indices, query_label)
            tpr, fpr = calculate_tpr_fpr(query_index, retrieved_indices)
            end_time = time.time()
            execution_time = end_time - start_time
            results.append({
                "query_index": query_index, "precision": precision,
                "recall": recall, "f1_score": f1_score,
                "time": execution_time, "retrieved": retrieved_indices[:5]
            })
            tpr_avg += tpr
            fpr_avg += fpr
        if i % 20 == 19:
            print_results(results, bin, threshold)
        tprs.append(tpr_avg / 10)
        fprs.append(fpr_avg / 10)
    if i == 99:
        plot_all_queries(results, dataset_plot, bin)

roc_auc = auc(fprs, tprs)
print(f"..... AUC for {bin} bins: {roc_auc} .....")
plot_ROC(roc_auc, bin, fprs, tprs)

```

## 6.13. Appendix 13

```

folder = "dataset"
dataset_CV, dataset_plot = load_images_from_folder(folder)
query_indexes = [13, 134, 212, 320, 496, 500, 628, 738, 879, 941]
weights_list = [[1, 1, 1], [0.5, 0.25, 0.25], [0.25, 0.5, 0.25],
                [0.25, 0.25, 0.5], [0, 0, 0]]

for weight_index, weights in enumerate(weights_list):
    tprs = []

```

```

fprs = []
all_histograms = [extract_color_moments(img, weights, weight_index == 4) for img in dataset_CV]
max_threshold = find_max_threshold(query_indexes, all_histograms)
thresholds = np.linspace(0, max_threshold, 100)
for i, threshold in enumerate(thresholds):
    tpr_avg = 0
    fpr_avg = 0
    results = []
    for query_label, query_index in enumerate(query_indexes):
        query_hist = all_histograms[query_index]
        start_time = time.time()
        retrieved_indices = retrieve_images(query_hist, all_histograms, threshold)
        precision, recall, f1_score = compute_metrics(retrieved_indices, query_label)
        tpr, fpr = calculate_tpr_fpr(query_index, retrieved_indices)
        end_time = time.time()
        execution_time = end_time - start_time
        results.append({
            "query_index": query_index, "precision": precision,
            "recall": recall, "f1_score": f1_score,
            "time": execution_time, "retrieved": retrieved_indices[:5]
        })
        tpr_avg += tpr
        fpr_avg += fpr
    if i % 20 == 19:
        print_results(results, weight_index+1, threshold)
    tprs.append(tpr_avg / 10)
    fprs.append(fpr_avg / 10)
    if i == 99:
        plot_all_queries(results, dataset_plot, weight_index+1)

roc_auc = auc(fprs, tprs)
print(f".....: AUC for weights {weight_index+1}: {roc_auc} .....")
plot_ROC(roc_auc, weight_index+1, fprs, tprs)

```

## 6.14. Appendix 14

```

folder = "dataset"
dataset_CV, dataset_plot = load_images_from_folder(folder)
query_indexes = [13, 134, 212, 320, 496, 500, 628, 738, 879, 941]

tprs = []
fprs = []
all_histograms = [extract_lbp_features(img) for img in dataset_CV]
max_threshold = find_max_threshold(query_indexes, all_histograms)
thresholds = np.linspace(0, max_threshold, 100)
for i, threshold in enumerate(thresholds):
    tpr_avg = 0
    fpr_avg = 0
    results = []
    for query_label, query_index in enumerate(query_indexes):
        query_hist = all_histograms[query_index]

```

```

start_time = time.time()
retrieved_indices = retrieve_images(query_hist, all_histograms, threshold)
precision, recall, f1_score = compute_metrics(retrieved_indices, query_label)
tpr, fpr = calculate_tpr_fpr(query_index, retrieved_indices)
end_time = time.time()
execution_time = end_time - start_time
results.append({
    "query_index": query_index, "precision": precision,
    "recall": recall, "f1_score": f1_score,
    "time": execution_time, "retrieved": retrieved_indices[:5]
})
tpr_avg += tpr
fpr_avg += fpr
if i % 20 == 19:
    print_results(results, threshold)
tprs.append(tpr_avg / 10)
fprs.append(fpr_avg / 10)
if i == 99:
    plot_all_queries(results, dataset_plot)

roc_auc = auc(fprs, tprs)
print(f"::::::::::::: AUC: {roc_auc} :::::::::::::::")
plot_ROC(roc_auc, fprs, tprs)

```