BIRZEIT UNIVERSITY

COMPUTER VISION ENCS5343.

ASSIGNMENT#2.

PREPARED BY:

SALEH KHATIB – 1200991.

SECTION 2.

DR: AZIZ QAROUSH.

DATE: 8/1/2024.

Table of Contents

Table of figures

Introduction

CBIR

Content-based image retrieval (CBIR) is a method that searches digital images based on their content in large databases using computer vision techniques. Instead of examining an image's metadata, such as keywords, tags, or description, CBIR examines the image's content. In this context, “content” can refer to any information that can be inferred from the image itself, including colors, shapes, textures, and many other details. Since metadata-only searches rely on keyword completeness and quality, CBIR is preferred. To identify the most similar images, CBIR compares the content of the input image with images in the database. For CBIR to perform well in retrieval, efficient feature representation and similarity metrics are essential.

Color histogram

The distribution of colors in an image is shown through a color histogram. This statistic can be considered an approximation of the continuous distribution of color values that underlie it. A color histogram in digital images shows how many pixels in an image's color space—the set of all possible colors—that contain colors in each of a list of predefined color ranges. Although the term "color histogram" is frequently associated with 3D spaces such as RGB or HSV, it can be created for any type of color space. The term intensity histogram can be used alternatively for monochrome images. The color histogram values come from statistics. They show the statistical distribution of colors and the essential tone of an image.

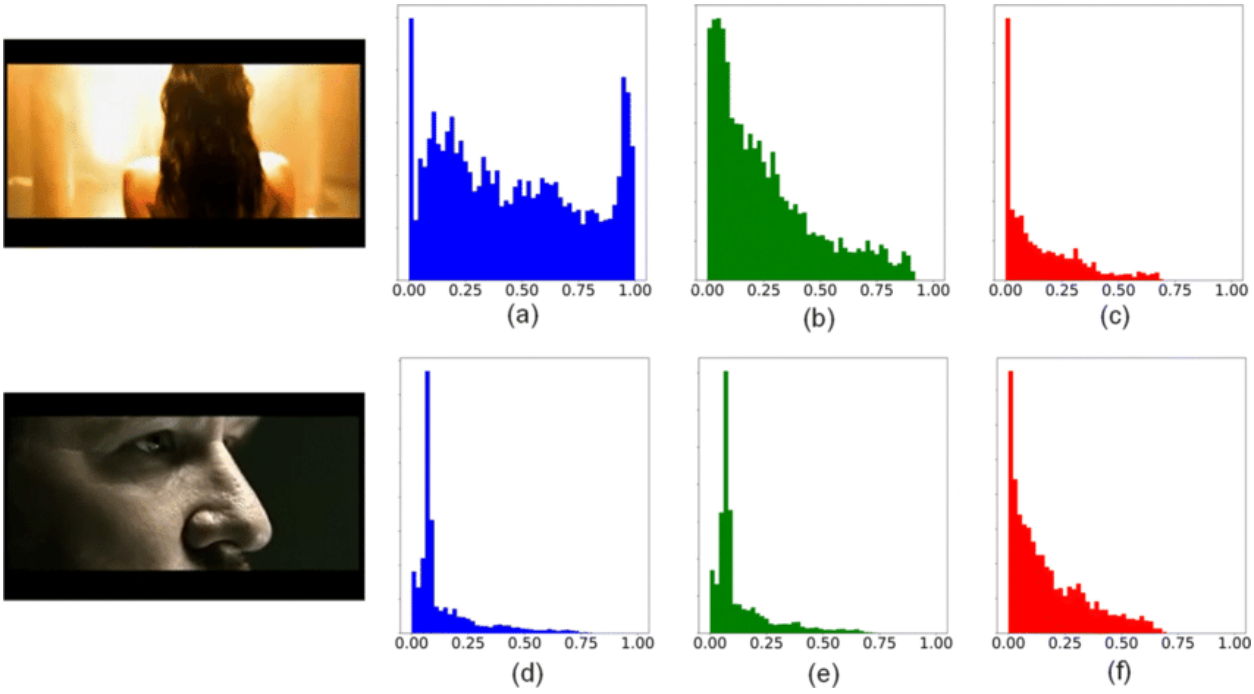

Figure 1: color histogram example.

Color moments

Color moments are measures that characterize color distribution in an image in the same way that central moments uniquely describe a probability distribution. They are mainly used for color indexing purposes as features in image retrieval applications to compare how similar two images are based on color. Color moments are scaling and rotation invariant, and can be computed for any color model. The first three color moments are usually used as features in image retrieval applications as most of the color distribution information is contained in the low-order moments. Since color moments encode both shape and color information, they are a good feature to use under changing lighting conditions, but they cannot handle occlusion very successfully.

The first color moment can be interpreted as the average color in the image, and it can be calculated by using the following formula where N is the number of pixels in the image and is the value of the j-th pixel of the image at the i-th color channel:

$$\mu i=N1j=1\sum NIj,i$$

The second color moment is the standard deviation, which is obtained by taking the square root of the variance of the color distribution:

$$\sigma i=N1j=1\sum N(Ij,i-\mu i)2$$

The third color moment is the skewness. It measures how asymmetric the color distribution is, and thus it gives information about the shape of the color distribution:

$$\gamma i=\sigma i3N1\sum j=1N(Ij,i-\mu i)3$$

Kurtosis is the fourth color moment, and, similarly to skewness, it provides information about the shape of the color distribution. More specifically, kurtosis is a measure of how extreme the tails are in comparison to the normal distribution.

Higher-order color moments are usually not part of the color moments feature set in image retrieval tasks as they require more data in order to obtain a good estimate of their value, and also the lower-order moments generally provide enough information.

System Implementation

The architecture of my CBIR system is as follow:

1- Image Preprocessing.
2- Feature extraction and Calculation.
3- Feature Representation.
4- Database or files Construction.
5- Query Image Processing.
6- Feature extraction and Calculation for Query Image.
7- Similarity and distance Measurement.
8- Ranking and Retrieval.
9- Presentation of Results.

From this architecture I build my main CBIR for different image and color representation which are: color histogram for different number of pins, color moments, and Histogram of Oriented Gradients (HOG).

To build my CBIR, I use python for programing, which is easy to write and have many ABI's and libraries which are:

```
import cv2
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from scipy.stats import skew, mode, kurtosis
from scipy.spatial import distance
from skimage.feature import hog
import time
```

Figure 2: API's.

Form the figure2 I use:

1- cv2(OpenCV).
2- Numpy.
3- PIL (Python Imaging Library).
4- Matplotlib.

5- Scipy.
6- Skimage.
7- Time.

Now we will see how I program it.

```python
def imgeload(imgnum):
    img = cv2.imread("Images/"+str(imgnum)+".jpg")
    img2 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    return img2

def color_hist(img,b):
    fet = []
    for i in range(3):
        histo = cv2.calcHist([img], [i], None, [b], [0, 256])
        nhisto = histo / histo.sum()
        fet.append(nhisto)

    feat_vec = np.concatenate((fet[0].flatten(),fet[1].flatten(),fet[2].flatten()))
    return feat_vec

def color_mom(img):
    mean = np.mean(img)
    std = np.std(img)
    skewness = skew(img.flatten())
    feat_vec = np.array([mean,std,skewness])
    return feat_vec

def color_mom2(img):
    mean = np.mean(img)
    std = np.std(img)
    skewness = skew(img.flatten())
    med = np.median(img)
    mode_v = mode(img.reshape(-1,3),axis=None)
    kur = kurtosis(img.flatten())
    feat_vec = np.array([mean,std,skewness,med,mode_v[0],mode_v[1],kur])
    return feat_vec

def color_hist_HOG(img):
    fet = []
    for i in range(3):
        histo = cv2.calcHist([img], [i], None, [120], [0, 256])
        nhisto = histo / histo.sum()
        fet.append(nhisto)

    his_vec = np.concatenate((fet[0].flatten(),fet[1].flatten(),fet[2].flatten()))
    imggry = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    hog_fet = hog(imggry, orientations=8, pixels_per_cell=(16, 16),cells_per_block=(1, 1), block_norm='L2-Hys')
    nhis_vec = his_vec / np.linalg.norm(his_vec)
    nhog_fet = hog_fet / np.linalg.norm(hog_fet)
    feat_vec = np.concatenate((nhis_vec,nhog_fet))
    return feat_vec
```

Figure 3: CPIR programing 1.

As we see in figure3, I split my code to functions, to be easy to understand and deal with the system. So, my functions are:

imgeload: which is function that take the path of the image and return the image as NDarray which I convert the image to RGB space.

color_hist: function that take image and the number of pins, then it returns feature vector which is the normalize color histogram for every channel concatenate together.

color_mom: function that take image, then it returns feature vector which is the color moments (mean, std, skewness) for the image.

color_mom2: function that take image, then it returns feature vector which is the color moments (mean, std, skewness, median, mode, kurtosis) for the image.

color_hist_HOG: function that take image, then it returns feature vector which is the normalize color histogram for every channel concatenate together and normalize HOG features.

```python
def His_to_str(imgnum,vec):
    strres = ""
    for i in range(len(vec)-1):
        strres += str(vec[i]) + ","

    strres += str(vec[-1]) + "_" + str(imgnum) +"\n"

    return strres

def train(fi,op,b):
    f = open(fi,"w")
    for i in range(1000):
        img = imgeload(i)
        if(op == "1"):
            feat_vec = color_hist(img,b)
        elif(op == "2"):
            feat_vec = color_mom(img)
        elif(op == "3"):
            feat_vec = color_mom2(img)
        else:
            feat_vec = color_hist_HOG(img)
        s = His_to_str(i,feat_vec)
        f.write(s)

def CBIR(img_name,fi,op,b,worn):
    w = np.array([1,0.5,10])
    w2 = np.array([1,0.5,10,1,2,2,3])
    img = cv2.imread(img_name)
    img2 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    if(op == "1"):
        vec_tst = color_hist(img2,b)
    elif(op == "2"):
        vec_tst = color_mom(img2)
    elif(op == "3"):
        vec_tst = color_mom2(img2)
    else:
        vec_tst = color_hist_HOG(img2)
    fr = open(fi,"r")
    dis_res = []
    for x in fr:
        x_split = x.split("_")
        nums = x_split[0].split(",")
        numsnp = np.array(nums)
        vec = numsnp.astype(np.float32)
        if(worn == 1):
            if(op == "2"):
                E_D =  distance.euclidean(vec_tst*w, vec*w)
            elif(op == "3"):
```

Figure 4: CPIR programing 2.

His_to_str: this function take image number and feature vector, and return these inputs as string.

Train: this function takes file name, operation (color histogram, moments ...), and number of pins. This function calculates the feature vectors for all the data set and save it on a file.

CBIR: this function takes image name, file name, operation (color histogram, moments ...), number of pins, and (worn) which mean weighted or not. This function calculates the feature vector for the test image and compete the Euclidean distance to all the database, and return distance list.

```
def truep (numimg):

    hun = numimg // 100
    res = np.arange(hun*100, (hun+1)*100)
    return res

def metrics(dist,thre,numimg):
    re_image = np.where(dist <= thre)[0]
    all_rele = len(re_image)
    pos = truep(numimg)

    true_pos = len(set(re_image).intersection(pos))
    fp = all_rele - true_pos
    tn = len(dist) - all_rele - (len(pos)-true_pos)
    fpr = fp / (fp + tn) if (fp + tn) > 0 else 0
    precision = true_pos / all_rele if all_rele > 0 else 0
    recall = true_pos / len(pos) if len(pos) > 0 else 0
    f1_score = (2 * precision * recall) / (precision + recall) if (precision + recall) > 0 else 0
    return precision ,recall, f1_score, fpr

def ROC(qur,file,op,b,worn):
    start_time = time.time()

    psa = [0] * 100
    rsa = [0] * 100
    fsa = [0] * 100
    fpr = [0] * 100
    fig, axs = plt.subplots(3, 4, figsize=(12, 8))
    AUC_VAL = []
    for q, ax in zip(quries,axs.flat):
        dis = CBIR("test/"+str(q)+".jpg",file,op,b,worn)

        PS = []
        RS = []
        FS = []
        fpl = []
        thr_val = np.linspace(min(dis),max(dis), num=100)
        for thre in thr_val:
            p,r,f,fp = metrics(dis,thre,q)
            PS.append(p)
            RS.append(r)
            FS.append(f)
            fpl.append(fp)

        psa[:] = [x + y for x, y in zip(psa, PS)]
        rsa[:] = [x + y for x, y in zip(rsa, RS)]
        fpr[:] = [x + y for x, y in zip(fpr, fpl)]
        fsa.append(sum(FS) / len(FS))
```

Figure 5: CPIR programing 3.

Truep: a function that Specifies the image category.

Metrics: a function that take the distance list, threshold, and image number, and return precision, recall, f1_score, and FPR.

ROC: this function takes the query list, file name, operation (color histogram, moments ...), number of pins, and (worn) which mean weighted or not. This function draw ROC, and calculate all average metrics.

```
def Show_Img(dis_resnp):
    ret_inx = np.argpartition(dis_resnp, 10)[:10]
    fig, axes = plt.subplots(nrows=2, ncols=5, figsize=(12, 6))

    #Loop through image files and display each image
    j = 1
    for i, ax in enumerate(axes.flat):
        img = mpimg.imread("Images/"+str(ret_inx[i])+".jpg")
        ax.imshow(img)
        ax.axis('off')  # Hide axis labels
        ax.set_title('Image_num: '+ str(j))
        j += 1

# Show the plot with ten images
    plt.tight_layout()
    plt.show()


quries = [16,135,221,319,465,585,649,738,871,991]


print("Welcome to my CBIR\n")

while(1):
    print("what do you want: 1-) train or 2-) CBIR or 3-) test")
    print("end for END")
    ch = input("put your choice: ")

    if(ch == "1"):

        print("choose the algorithm: ")
        print("1-) color histogram")
        print("2-) color moments1")
        print("3-) color moments2")
        print("4-) color histogram + HOG type")
        op = input("put your choice: ")

        if(op == "1"):
            print("how many bins do you want?")
            print("a. 120")
            print("b. 180")
            print("c. 360")
            op2 = input("put your choice: ")
            if(op2 == "a"):
                train("databaseCH120.txt","1",120)
                print("train is done")
            elif(op2 == "b"):
                train("databaseCH180.txt","1",180)
                print("train is done")
```

Figure 6: CPIR programing 4

Show_Img: this function takes the distance list, and show the most 10 images similar to our query image.

And the last thing is the menu for choosing what do you want.

Experimental Setup and results

evaluation methodology

First, I use Wang datasets, because in has 1000 image distributed over 10 categories, which when I want to calculate the true positive it will be easy, by just knowing the number of the image.

Then my evaluation metrics are: precision, recall, f1_score, testing time, AUC, ROC curve, and FPR (False positive rate).

But how I can do it?

It simple, as we see in the setup, I test 10 different images, for every image I find the distance list based on the image representation and its database, then I calculate the metrics for 100 thresholds, distributed between the min and max in distance list. After that, I draw the ROC curve and calculate the AUC. Finally, I calculate the average for all these query images and show the final result.

For showing the retrieval images, there is a function that show the most 10 images similar to our query image.

Results And Discussion

Color histogram

Now we will show the results for Color histogram for different number of bins, 120,180,360.

```
what do you want: 1-) train or 2-) CBIR or 3-) test
end for END
put your choice: 1
choose the algorithm:
1-) color histogram
2-) color moments1
3-) color moments2
4-) color histogram + HOG type
put your choice: 1
how many bins do you want?
a. 120
b. 180
c. 360
put your choice: a
Elapsed time: 2.831917 seconds
train is done
what do you want: 1-) train or 2-) CBIR or 3-) test
end for END
put your choice: 3
choose the algorithm:
1-) color histogram 120 pin
2-) color histogram 180 pin
3-) color histogram 360 pin
4-) color moments1 wetighed
5-) color moments1 not
6-) color moments2 wetighed
7-) color moments2 not
8-) color histogram + HOG type
put your choice: 1
Average AUC: 0.7138144444444444
AVG F1:  0.19856984477827508
AVG Precision:  0.2937481722434127
AVG Recall:  0.76038
Elapsed time: 3.501337 seconds
```

Figure 7: Color histogram 120 pins results.

As we see, this representation has a good performance (AUC), and short time for the experiment. But, it's very bad in terms of F1-score

Now ROC:

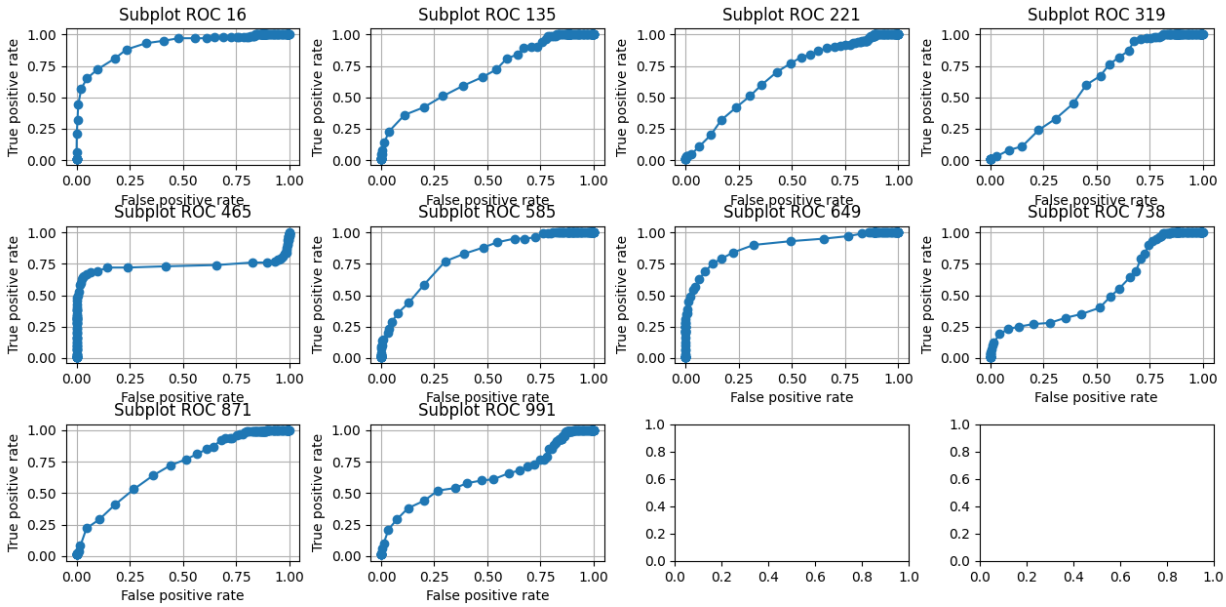

Figure 8: ROC for the 10 images CH120.

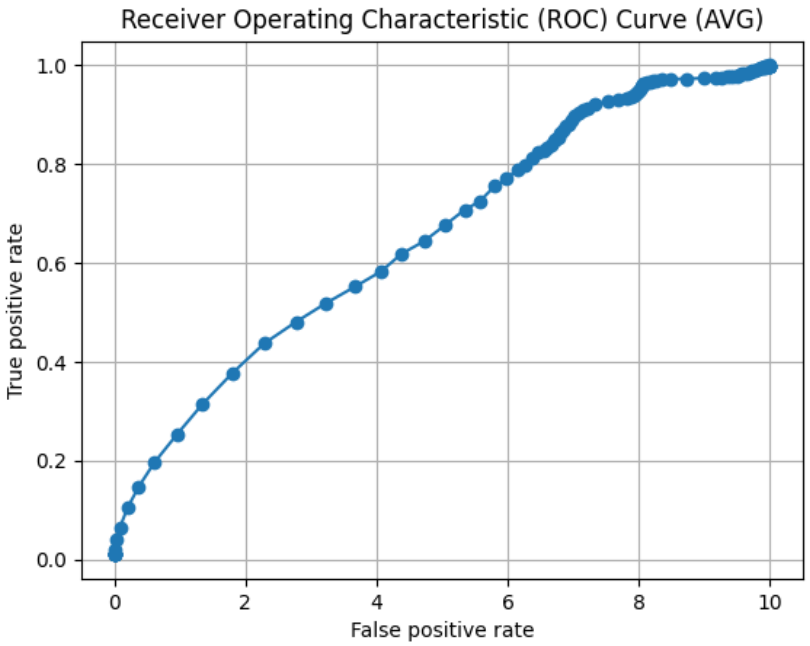

Figure 9: Avg ROC CH120.

Retrieval Results:

Note: I will use show_img function which will the top ten least distance, and you must know the first image is the test image:

Figure 10: top ten least distance images CH120.

Now for 180 pins:

```
what do you want: 1-) train or 2-) CBIR or 3-) test
end for END
put your choice: 1
choose the algorithm:
1-) color histogram
2-) color moments1
3-) color moments2
4-) color histogram + HOG type
put your choice: 1
how many bins do you want?
a. 120
b. 180
c. 360
put your choice: b
Elapsed time: 3.711192 seconds
train is done
what do you want: 1-) train or 2-) CBIR or 3-) test
end for END
put your choice: 3
choose the algorithm:
1-) color histogram 120 pin
2-) color histogram 180 pin
3-) color histogram 360 pin
4-) color moments1 wetighed
5-) color moments1 not
6-) color moments2 wetighed
7-) color moments2 not
8-) color histogram + HOG type
put your choice: 2
Average AUC: 0.6893083333333333
AVG F1:  0.19314158302277434
AVG Precision:  0.27454980029222287
AVG Recall:  0.7756899999999999
Elapsed time: 5.872282 seconds
```

Figure 11: Color histogram 180 pins results.

As we see, this representation has a good performance (AUC), and short time for the experiment, but bigger than 120 pin color histogram. But, it's very bad in terms of F1-score

Now ROC:

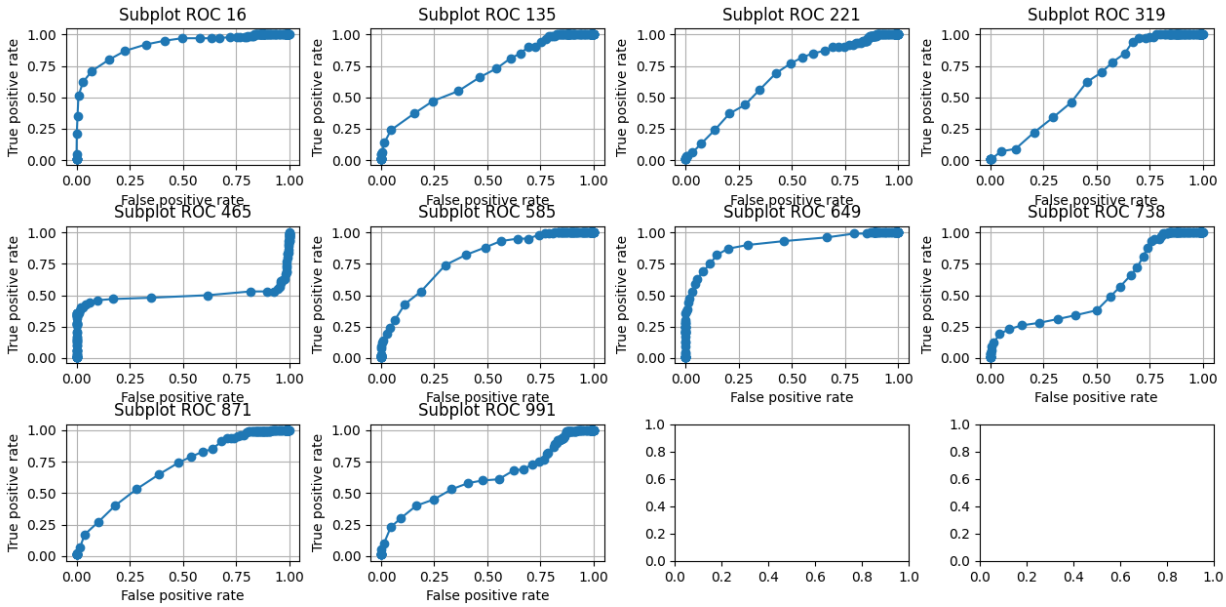

Figure 12: ROC for the 10 images CH180.

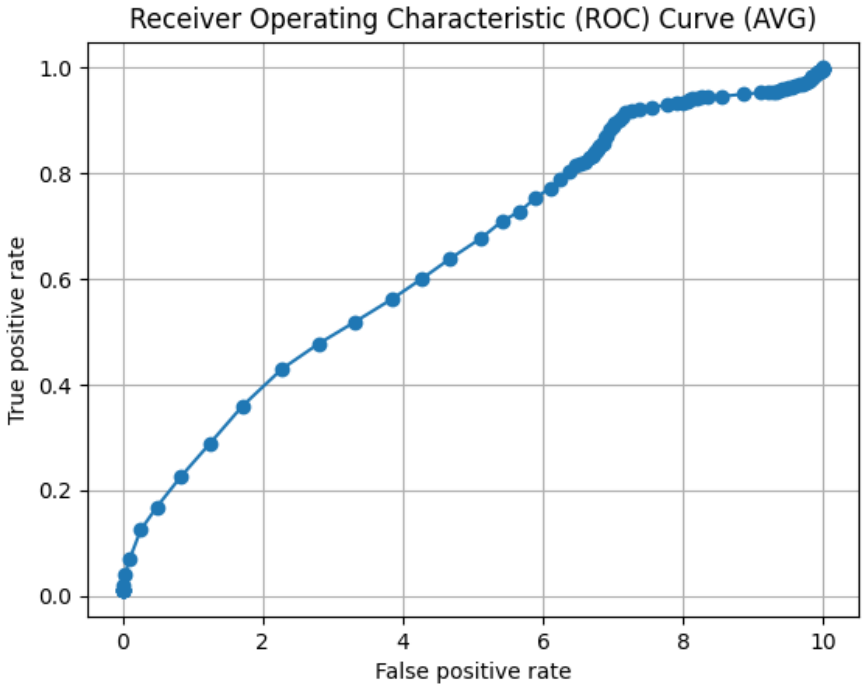

Figure 13: Avg ROC CH180.

Retrieval Results:

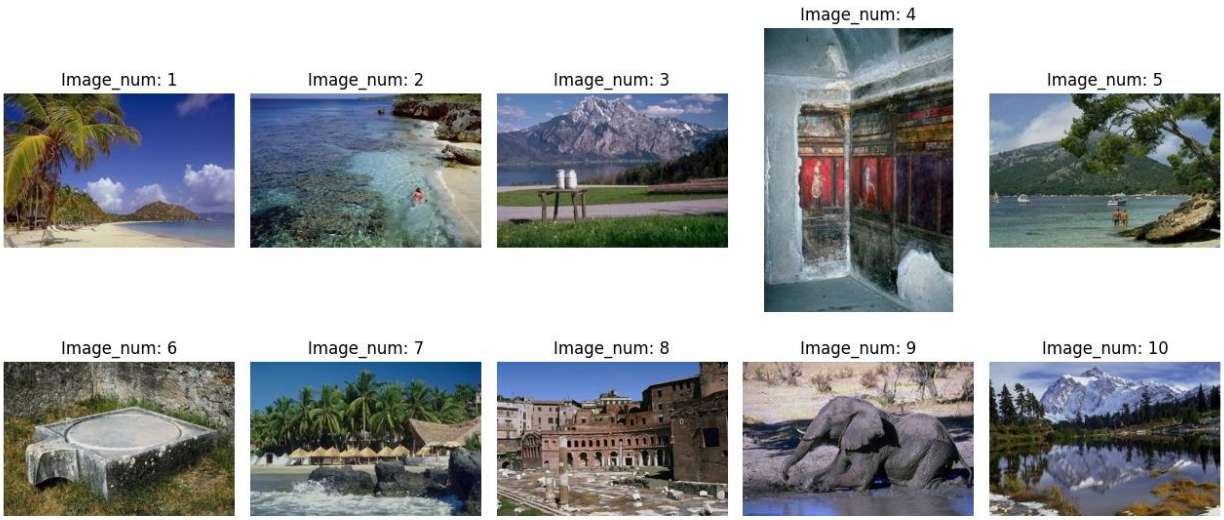

Figure 14: top ten least distance images CH180.

Now for 360 pins:

```
what do you want: 1-) train or 2-) CBIR or 3-) test
end for END
put your choice: 1
choose the algorithm:
1-) color histogram
2-) color moments1
3-) color moments2
4-) color histogram + HOG type
put your choice: 1
how many bins do you want?
a. 120
b. 180
c. 360
put your choice: c
Elapsed time: 2.573680 seconds
train is done
what do you want: 1-) train or 2-) CBIR or 3-) test
end for END
put your choice: 3
choose the algorithm:
1-) color histogram 120 pin
2-) color histogram 180 pin
3-) color histogram 360 pin
4-) color moments1 wetighed
5-) color moments1 not
6-) color moments2 wetighed
7-) color moments2 not
8-) color histogram + HOG type
put your choice: 3
Average AUC: 0.6848166666666666
AVG F1:  0.1898257482678008
AVG Precision:  0.2513875875595069
AVG Recall:  0.8019399999999999
Elapsed time: 9.370357 seconds
```

Figure 15: Color histogram 360 pins results.

As we see, this representation has a good performance (AUC), and short time for the experiment, but bigger than 120 and 180 pin color histogram. But, it's very bad in terms of F1-score

Now ROC:

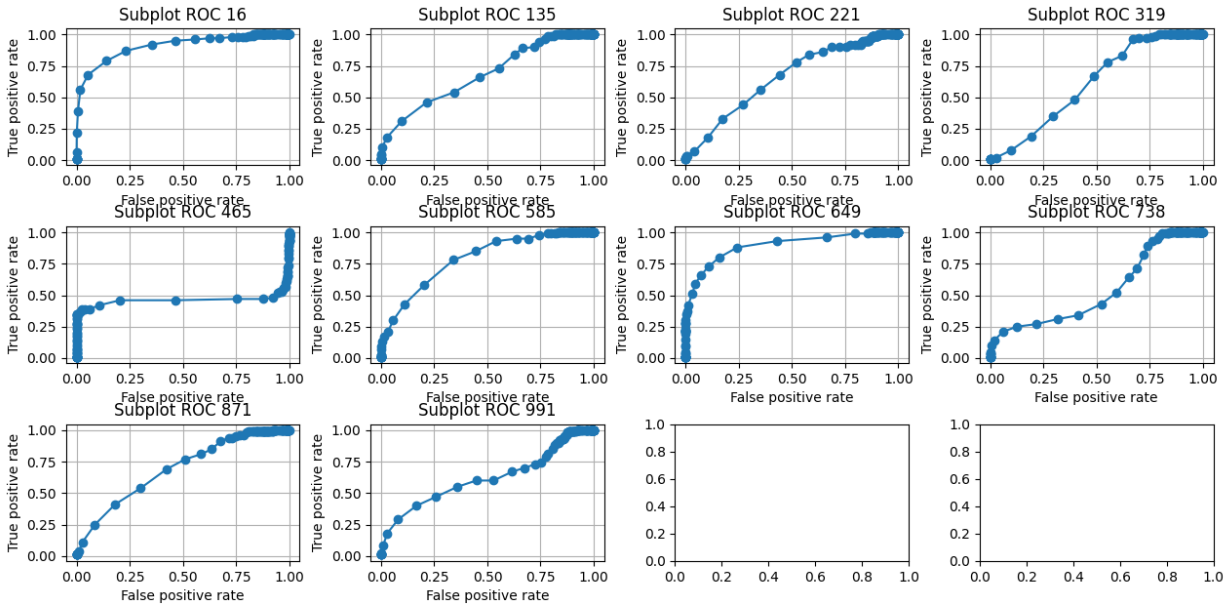

Figure 16: ROC for the 10 images CH360.

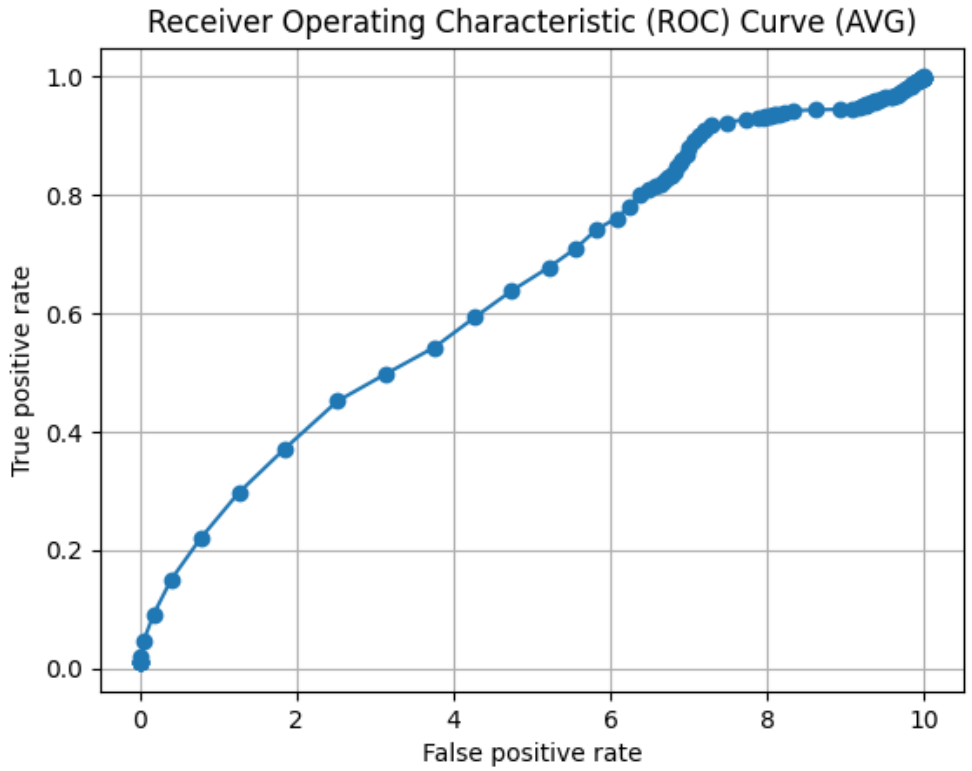

Figure 17: Avg ROC CH360.

Retrieval Results:

Figure 18: top ten least distance images CH360.

As we see, in terms of performance, execution time, precision, and F1-score, 120 pins color histogram is better, but for recall, as we increase the number of pins the recall increase.

Color moments

Now we will show the results for Color moments for different types which are: same weight color moments1 (mean, std, skew), different weight color moments1 (mean, std, skew), same weight color moments2 (mean, std, skew, median, mode, kurtosis), different weight color moments2 (mean, std, skew, median, mode, kurtosis).

Now we will start with CM1W (color moments1 different weight):

```
what do you want: 1-) train or 2-) CBIR or 3-) test
end for END
put your choice: 1
choose the algorithm:
1-) color histogram
2-) color moments1
3-) color moments2
4-) color histogram + HOG type
put your choice: 2
Elapsed time: 15.419410 seconds
train is done
what do you want: 1-) train or 2-) CBIR or 3-) test
end for END
put your choice: 3
choose the algorithm:
1-) color histogram 120 pin
2-) color histogram 180 pin
3-) color histogram 360 pin
4-) color moments1 wetighed
5-) color moments1 not
6-) color moments2 wetighed
7-) color moments2 not
8-) color histogram + HOG type
put your choice: 4
Average AUC: 0.7030144444444445
AVG F1:  0.23516792493735914
AVG Precision:  0.19720921394947097
AVG Recall:  0.7872699999999998
Elapsed time: 0.793719 seconds
```

Figure 19: CM1W results.

As we see, it takes a long time in training compared to the time in testing, this representation has a good performance (AUC), and short time for the experiment. But, it's very bad in terms of F1-score and precision.

ROC:

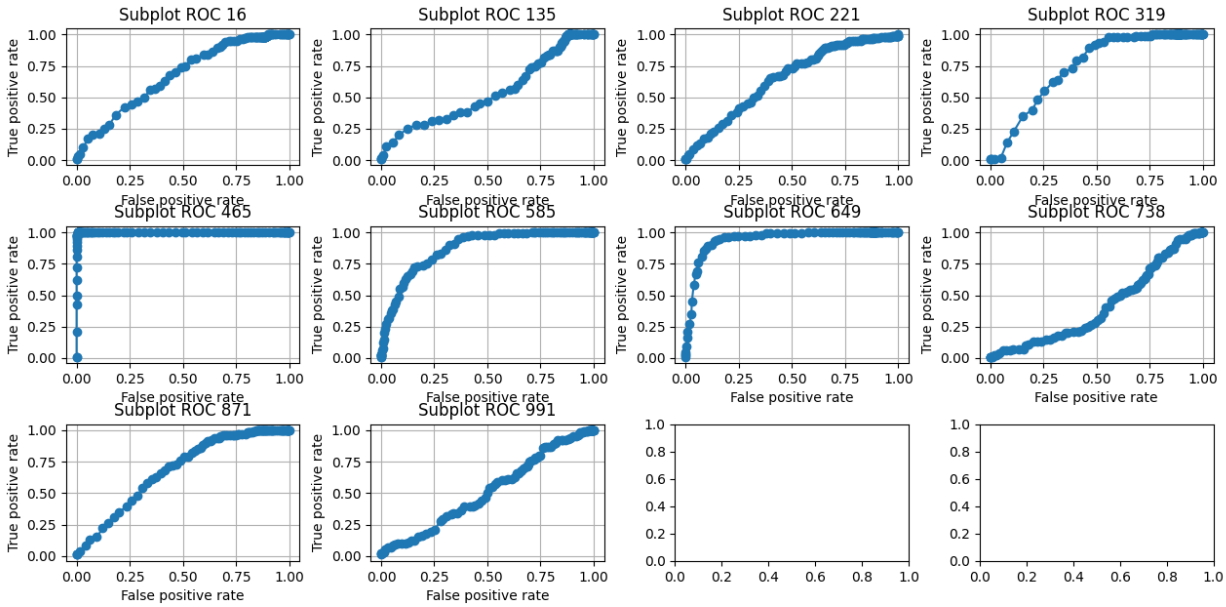

Figure 20: ROC for the 10 images CM1W.

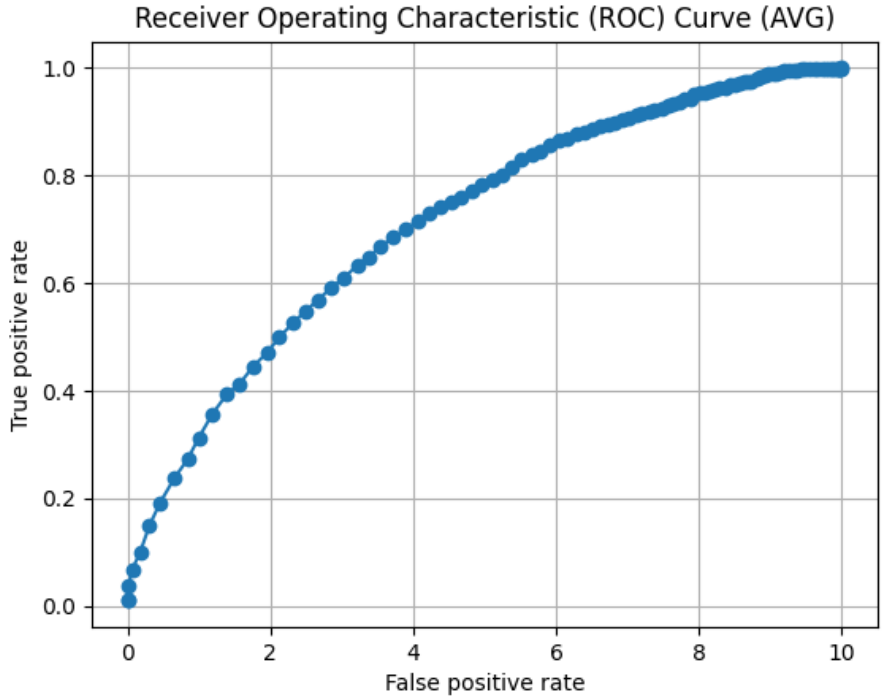

Figure 21: Avg ROC CM1W.

Retrieval Results:

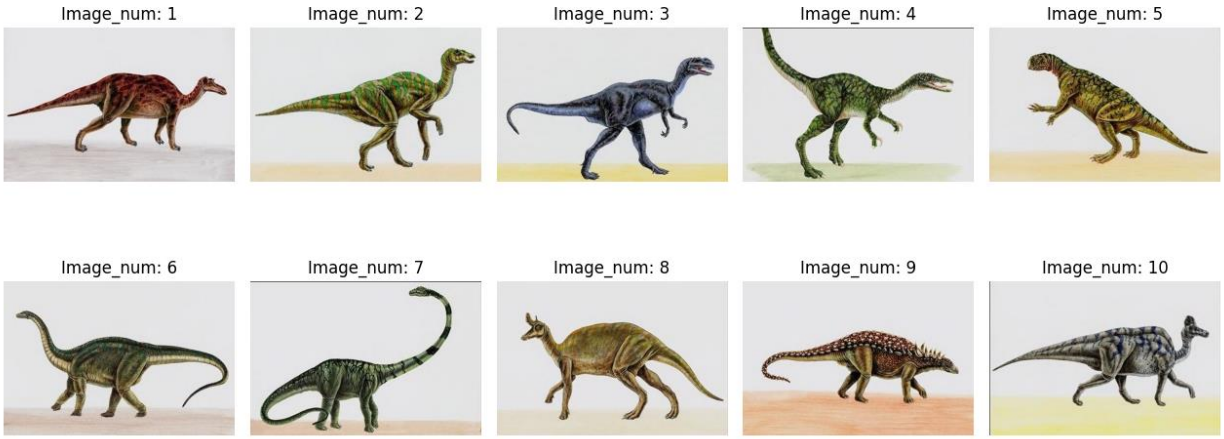

Figure 22: top ten least distance images CM1W.

Now CM1S:

```
what do you want: 1-) train or 2-) CBIR or 3-) test
end for END
put your choice: 3
choose the algorithm:
1-) color histogram 120 pin
2-) color histogram 180 pin
3-) color histogram 360 pin
4-) color moments1 wetighed
5-) color moments1 not
6-) color moments2 wetighed
7-) color moments2 not
8-) color histogram + HOG type
put your choice: 5
Average AUC: 0.7108388888888888
AVG F1:  0.23346031711554174
AVG tttttttt:  0.7712200000000002
AVG Precision:  0.1990379906250765
AVG Recall:  0.7712199999999996
Elapsed time: 0.502118 seconds
```

Figure 23: CM1S results.

As we note, I use the same database for CM1W. for testing time is less than the weighted one, because I didn't use weights, I assume that all weights equal one. This representation has a good performance (AUC), and short time for the experiment. But, it's very bad in terms of F1-score and precision.

ROC:

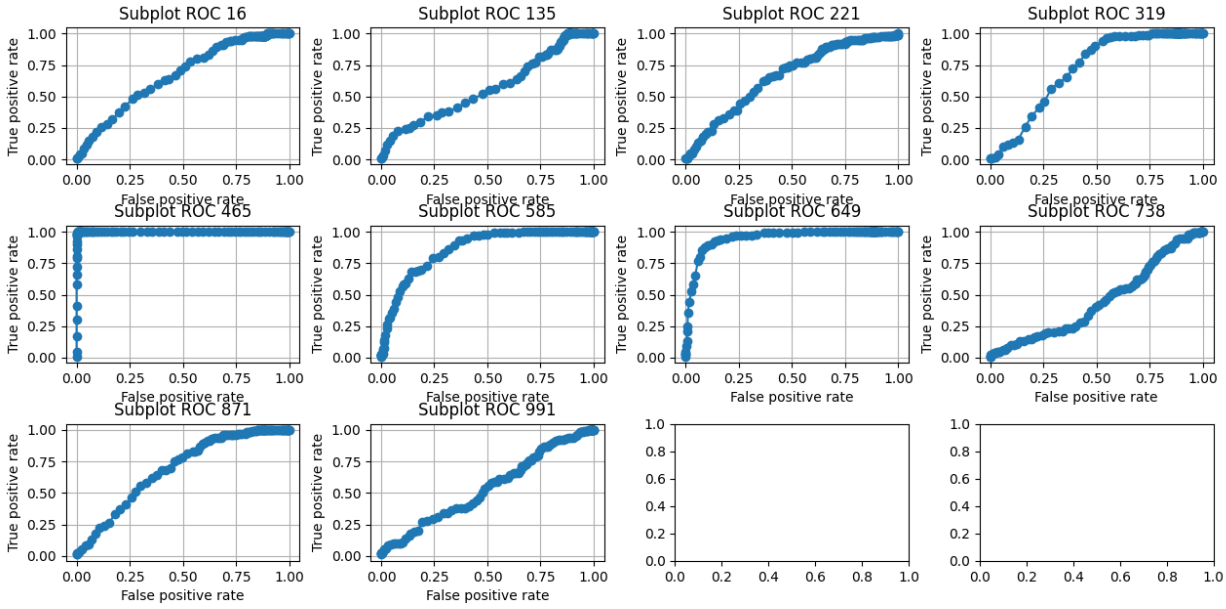

Figure 24: ROC for the 10 images CM1S.

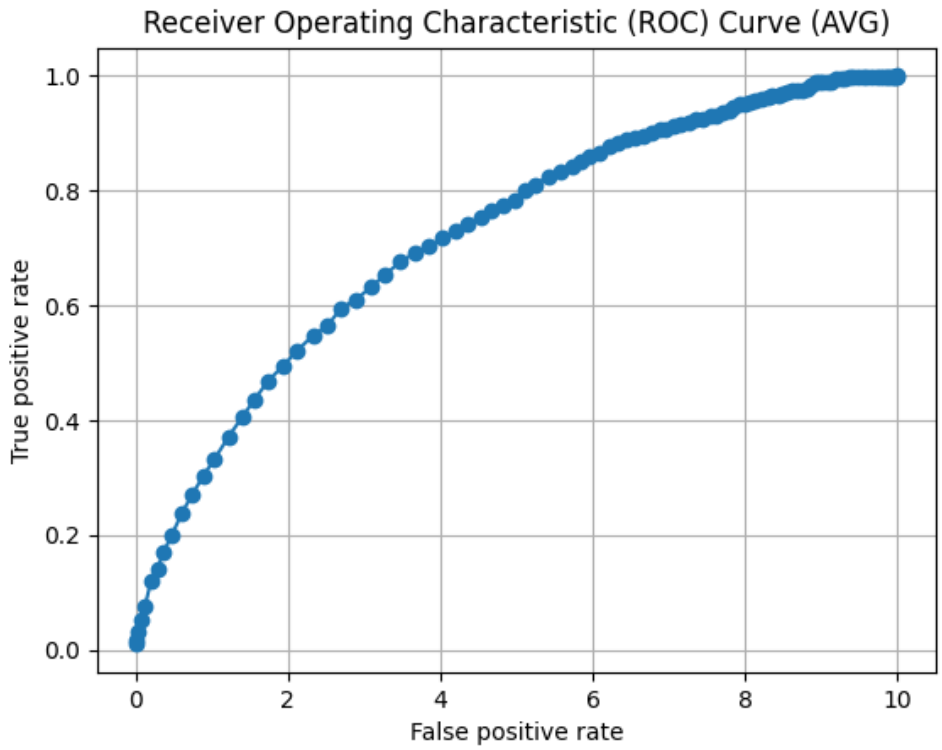

Figure 25: Avg ROC CM1S.

Retrieval Results:

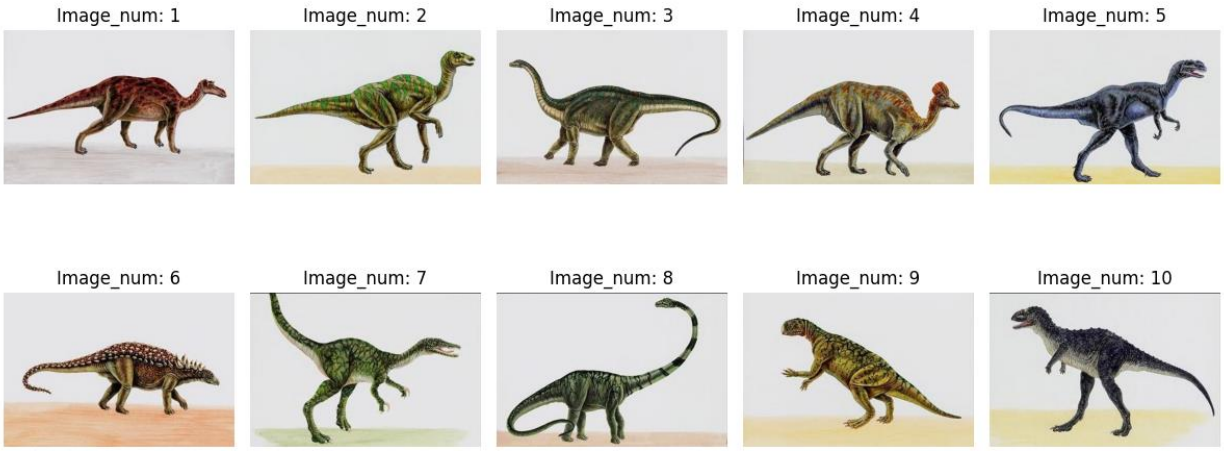

Figure 26: top ten least distance images CM1S.

Now CM2W:

```
what do you want: 1-) train or 2-) CBIR or 3-) test
end for END
put your choice: 1
choose the algorithm:
1-) color histogram
2-) color moments1
3-) color moments2
4-) color histogram + HOG type
put your choice: 3
Elapsed time: 30.829572 seconds
train is done
what do you want: 1-) train or 2-) CBIR or 3-) test
end for END
put your choice: 3
choose the algorithm:
1-) color histogram 120 pin
2-) color histogram 180 pin
3-) color histogram 360 pin
4-) color moments1 wetighed
5-) color moments1 not
6-) color moments2 wetighed
7-) color moments2 not
8-) color histogram + HOG type
put your choice: 6
Average AUC: 0.6138600000000001
AVG F1:  0.21217030147134208
AVG Precision:  0.16178361586454368
AVG Recall:  0.8784999999999998
Elapsed time: 1.144870 seconds
```

Figure 27: CM2W results.

As we see, it takes a long time in training compared to the time in testing and it twice the training time for CM1, due that has more moments. This representation has a good performance (AUC), and short time for the experiment. But, it's very bad in terms of F1-score and precision.

ROC:

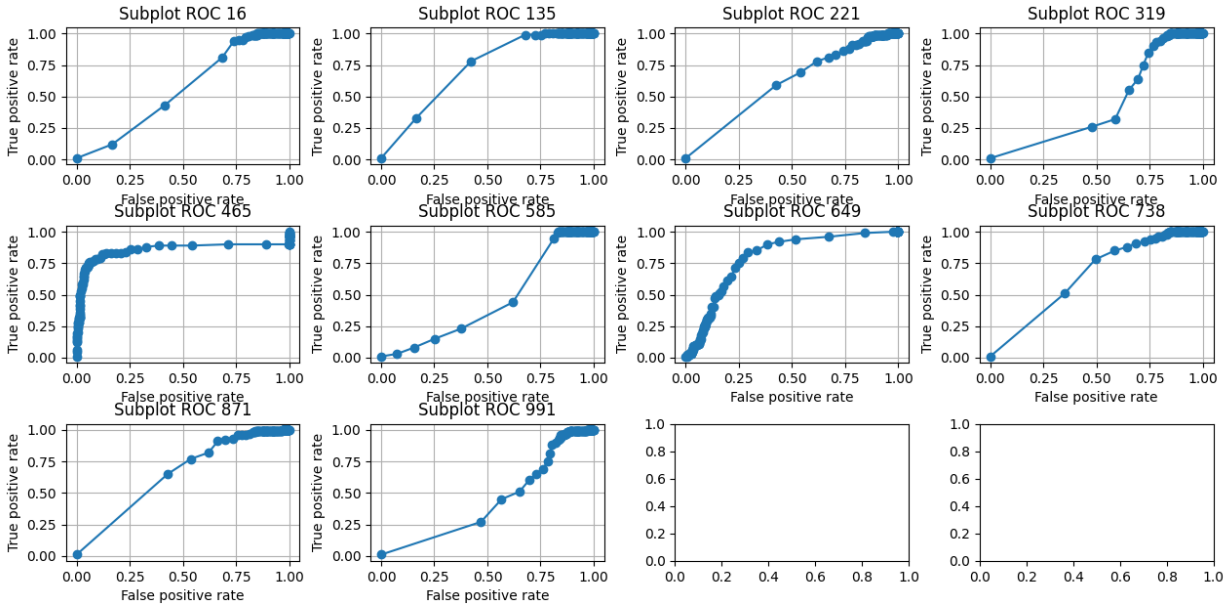

Figure 28: ROC for the 10 images CM2W.

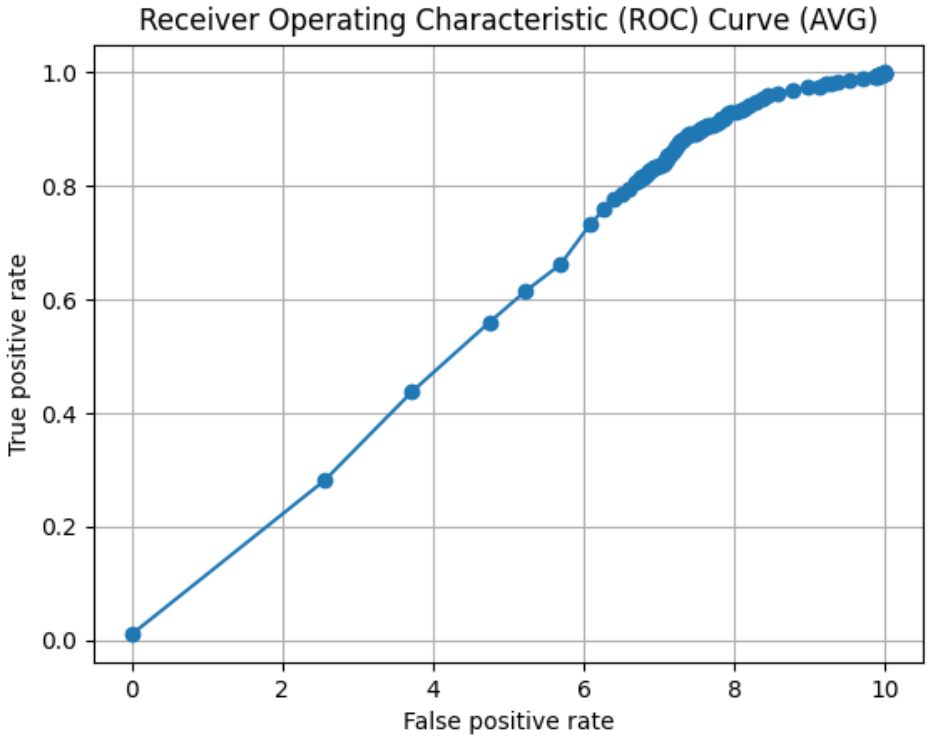

Figure 29: Avg ROC CM2W.

Retrieval Results:

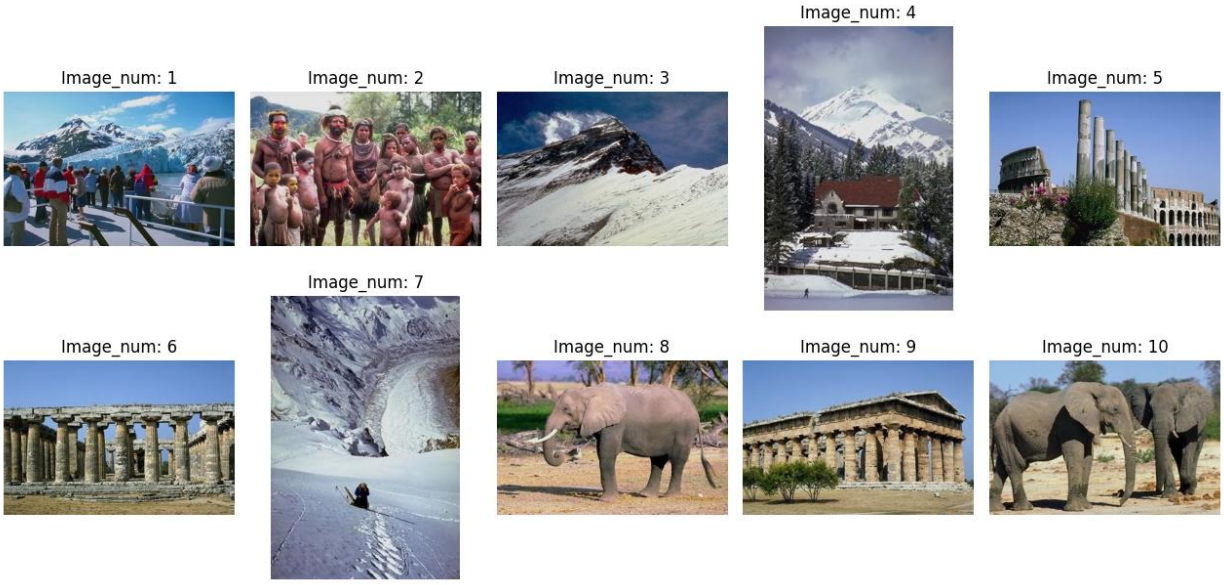

Figure 30: top ten least distance images CM2W.

Now CM2S:

```
what do you want: 1-) train or 2-) CBIR or 3-) test
end for END
put your choice: 3
choose the algorithm:
1-) color histogram 120 pin
2-) color histogram 180 pin
3-) color histogram 360 pin
4-) color moments1 wetighed
5-) color moments1 not
6-) color moments2 wetighed
7-) color moments2 not
8-) color histogram + HOG type
put your choice: 7
Average AUC: 0.6138605555555556
AVG F1:  0.2121704811658614
AVG Precision:  0.16181694919787704
AVG Recall:  0.8784999999999998
Elapsed time: 1.019654 seconds
```

Figure 31: CM2S results.

As we note, I use the same database for CM2W. This representation has a good performance (AUC), and short time for the experiment. But, it's very bad in terms of F1-score and precision.

ROC:

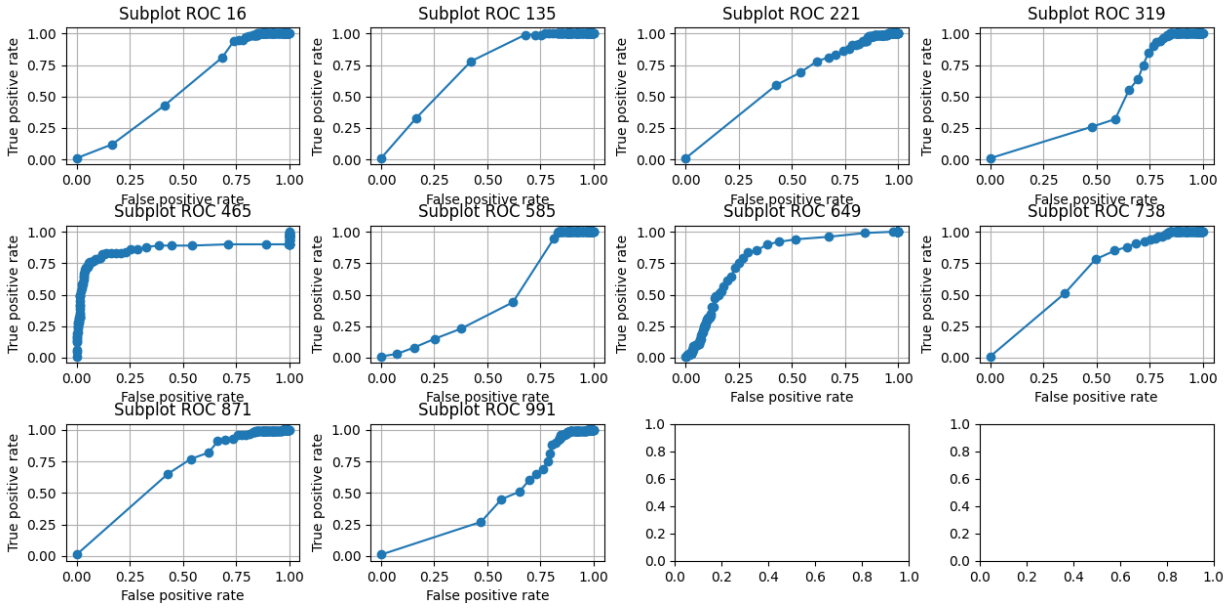

Figure 32: ROC for the 10 images CM2S.

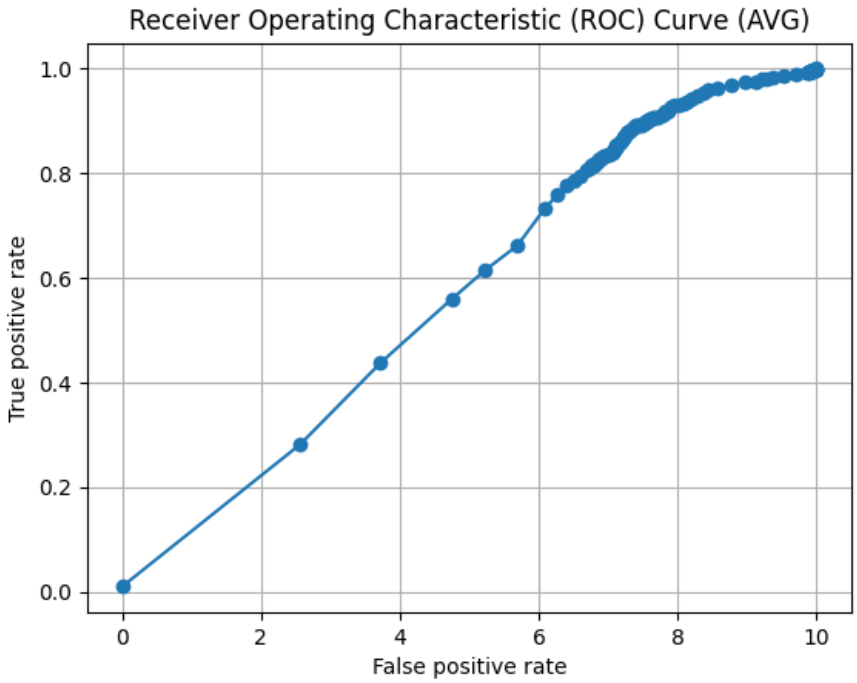

Figure 33: Avg ROC CM2S.

Retrieval Results:

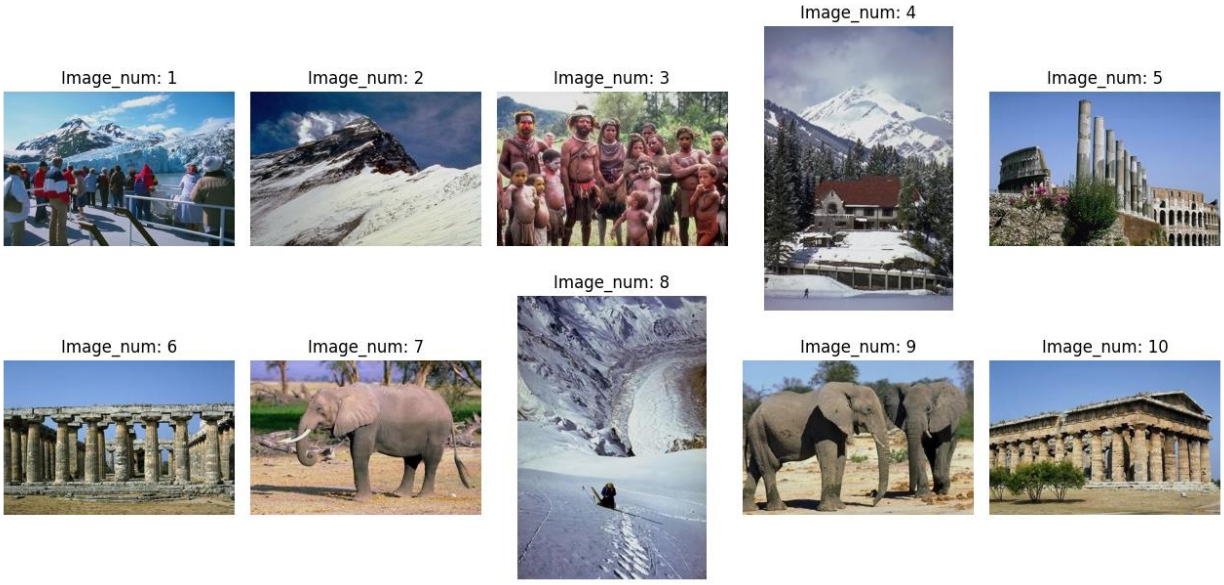

Figure 34: top ten least distance images CM2S.

As we note, changing the weights does not affect the CBIR, it will just decrease the precision and increase the recall a little bit.

For the moments we use, there is a difference, as we see the ROC is different from CM1 to CM2, for the training time as well, which for CM2 is twice the time for CM1, and even for the metrics, the recall in CM2 is better than CM1 and the opposite for precision.

Color histogram and HOG

Now we will improve our CBIR by adding HOG feature vectors for color histogram feature vectors.

```
what do you want: 1-) train or 2-) CBIR or 3-) test
end for END
put your choice: 1
choose the algorithm:
1-) color histogram
2-) color moments1
3-) color moments2
4-) color histogram + HOG type
put your choice: 4
Elapsed time: 40.766698 seconds
train is done
what do you want: 1-) train or 2-) CBIR or 3-) test
end for END
put your choice: 3
choose the algorithm:
1-) color histogram 120 pin
2-) color histogram 180 pin
3-) color histogram 360 pin
4-) color moments1 wetighed
5-) color moments1 not
6-) color moments2 wetighed
7-) color moments2 not
8-) color histogram + HOG type
put your choice: 8
Average AUC: 0.7496094444444444
AVG F1:  0.16555628713028123
AVG Precision:  0.5700316711550332
AVG Recall:  0.40342999999999996
Elapsed time: 66.486137 seconds
```

Figure 35: CHHOG results.

As we see, it takes a long time in training and testing, which is bad thing. For AUC it is the best from all other, and the most surprising thing is the precision and the recall, which the precision is better from all color features and bigger than recall, but for recall which is worst from all other color features.

ROC:

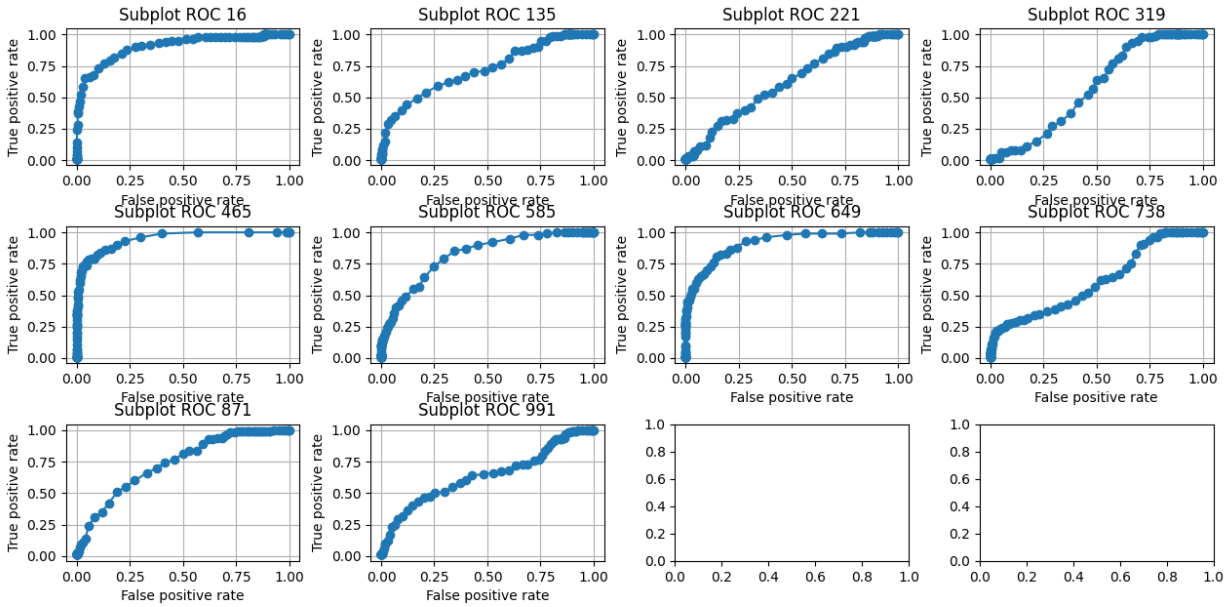

Figure 36: ROC for the 10 images CHHOG.

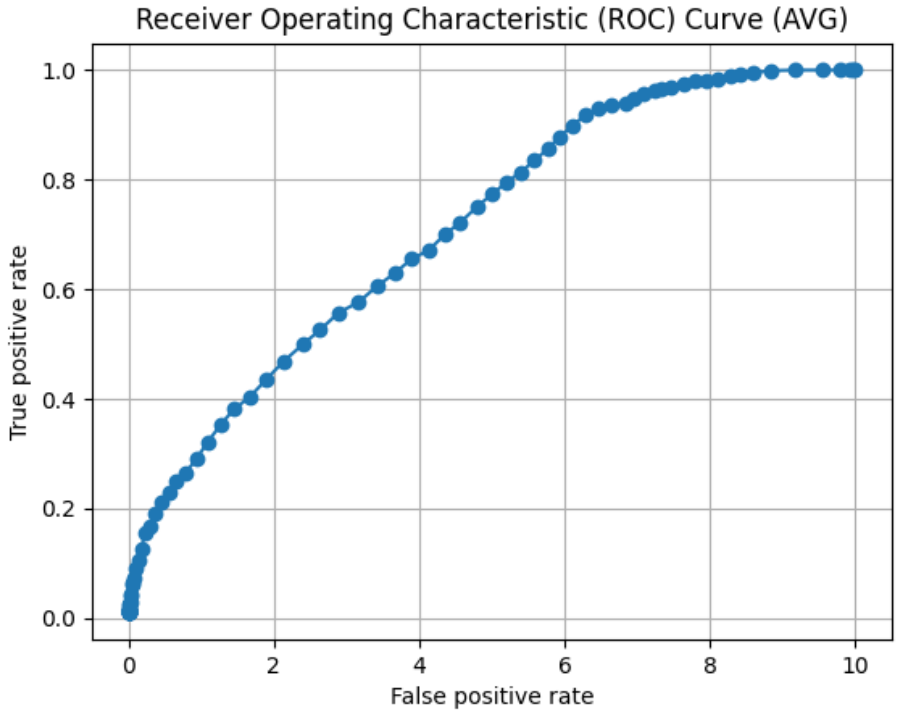

Figure 37: Avg ROC CHHOG.

Retrieval Results:

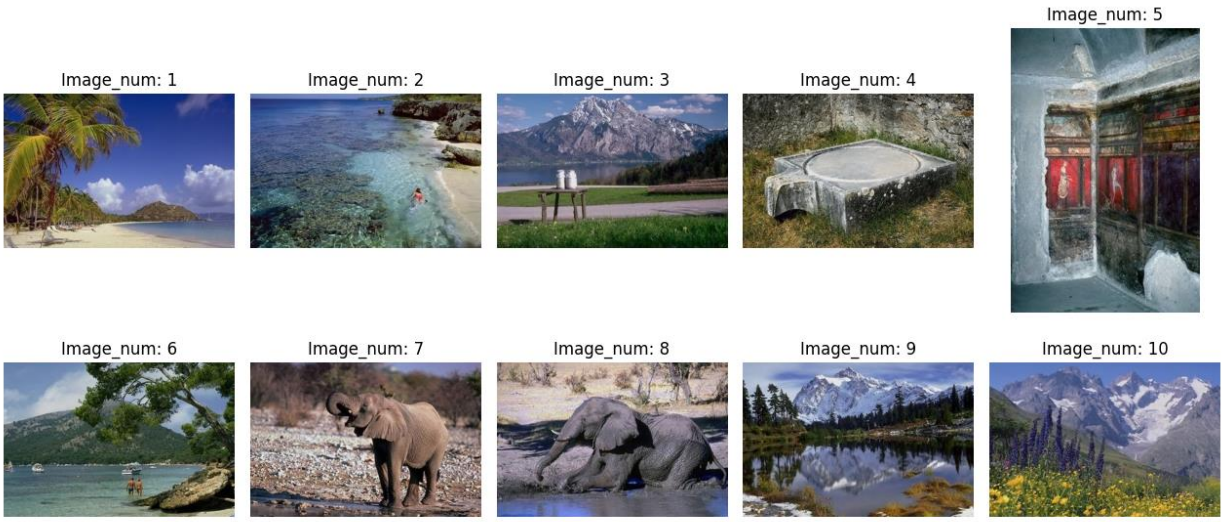

Figure 38: top ten least distance images CHHOG 1.

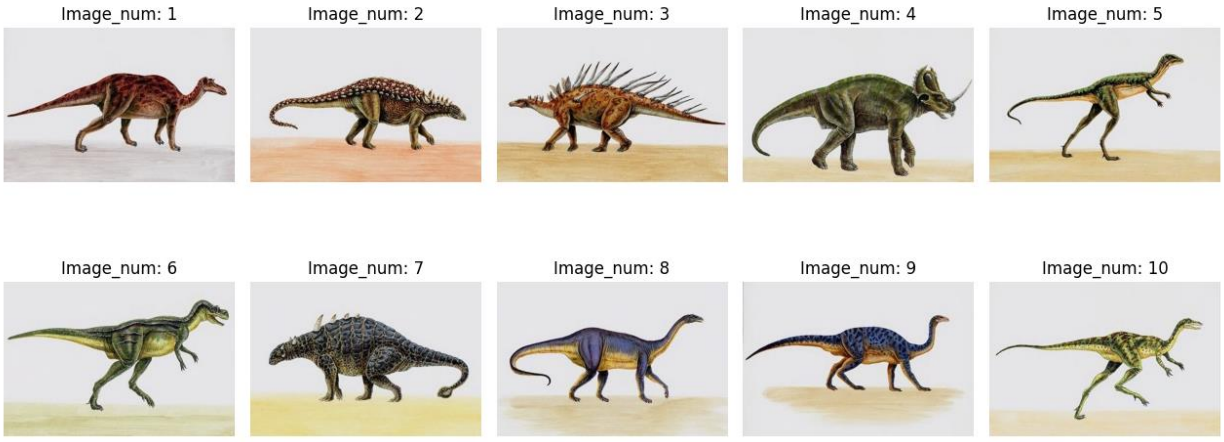

Figure 39: top ten least distance images CHHOG 2.

Figure 40: top ten least distance images CHHOG 3.

As we see, the system is good, but have some limitations:

1- The best average AUC is 0.75.
2- There is a tradeoff between recall and precision, so we can't have a best of two together.
3- In terms of f1-score is bad.
4- One of the biggest limitations is testing time, which will increase when the number of images and the size of feature vectors increase, so we that will make the time for retrieval the images will increase.

But we can improve the system by using other image features, also we can make the user the category of images they will search for, so that will make the time for retrieval process shorter, and choosing the best threshold.

Conclusion

To summarize, in this project we built a multi-method CBIR (content-based image retrieval) system with three main components: color histogram, color moment, and HOG (histogram of oriented gradations). Throughout this project, the process of exploration provided invaluable insights and learning opportunities. This has improved my ability to use OpenCV in Python, understand color features, use AUC (Area Under the Curve) and ROC (Receiver Operating Characteristics), calculate distances, create feature vectors, and easily create CBIR structures.

Individual evaluation of color features revealed unique attributes and performance indicators. Color histograms were found to be a straightforward but effective method, showing respectable AUC values of around 0.71 as well as moderate recall and bad precision rates. One of its noteworthy advantages was that it could be quickly tested and trained, which increased its practical applicability.

In a similar vein, color moments showed reasonable recall rates and bad precision rates and a reasonable level of simplicity, with AUC ranges from 0.61 to 0.71. Their proficiency on the test was beneficial even though they had longer periods of training.

Combining color histogram with HOG provided a more complex but promising approach. Despite displaying a commendable AUC of around 0.75, it has higher precision but relatively lower recall rates. However, this method required longer training and testing periods, which raised considerations regarding its practical implementation.

Regarding the overall effectiveness of color features in CBIR tasks, it is clear that each feature offers unique strengths and trade-offs. The simplicity of the color histogram and color moments facilitates ease of implementation and respectable performance, while including HOG enhances accuracy but requires more computational resources.

Future research and development should focus on improving hybrid feature methodologies, such as color histogram with HOG, in order to drive further progress in CBIR. This research should aim to achieve a balance between computational complexity and performance metrics. Furthermore, research into ways to incorporate new features, increase training effectiveness, and improve recall rates remain critical areas of development.

As a result, this project has enhanced the knowledge and application of CBIR systems while also pointing out directions for further research, and promoting the continued development of image retrieval and analysis techniques.

References

https://en.wikipedia.org/wiki/Content-based_image_retrieval

https://en.wikipedia.org/wiki/Color_histogram

course slides.