CS512: PROJECT REPORT
Content- Based Image Retrieval

Charu Saxena A20378393, Vishal Kumar Bimal A20380261
Department of Computer Science,Illinois Institute of Technology, Chicago,IL 60616, USA

ABSTRACT

This reports discusses about a method for content-based image retrieval(CBIR) by extracting features like color and texture. The retrieval experiments were performed by developing a database of images and querying the image to retrieve the set of similar images. First, images are defined in HSV color space. For color feature, localised histograms are made, and for texture, co-occurrence matrix were made for each image. During querying these features for images are compared to query image, and desired images are shown as a part of the result.

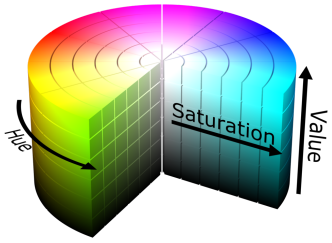
I. Introduction/Problem statement

In recent times, most of the data is coming in the form of images more than text and the size of the image are relatively more(a picture worth 60,000 words!), thus storage and transmission of a large number of images has become troublesome. Instead of text retrieval, image retrieval is wildly required in recent decades. Content-based image retrieval (CBIR) is regarded as one of the most effective ways of accessing visual data [2]. It deals with the image content itself such as color and texture structure instead of tags or annotated text. Huge amounts of data retrieval challenge the traditional database technology, but the traditional text-object database cannot satisfy the requirements of an image database.

This project is about, Content-based Image Retrieval(CBIR) , which aims in designing a system which can retrieve best matches for a queried image. The interest in CBIR has grown because of the limitations in metadata-based systems, as well as the large range of possible uses for efficient image retrieval. We can get textual information about images using tags and other existing technology, but this work involves manually describe each image in the database, which is tiring[1]. The efficiency of a CBIR depends largely on the features extracted from images to be used for matching purpose. The rest of the report is organised as follows. In section 2 we have discussed the proposed solution describing the underlying concept used and algorithm used to imply it. Section 3 deals with the implementation details, describing the program design issues, problems faced in it and function used with parameters. Section 4 describes the results and discussions of the experiment performed, comparing performances at different level and precision level for the result. It also discussed strengths and weakness of the program and effect of various parameters tuning described in section 3. The last section 5 is about conclusion, references and source code.

II. Proposed Solution

As discussed in the paper[1] there were ways to reduce semantic gap between the visual and human rich semantics. The image is first of all changes to HSV color space because just RGB color space fails to mimic how humans perceive color. Instead, we are going to use the HSV color space which maps pixel intensities into a cylinder[4] fig[i]



fig[i]

The most practical way include using the features of the image is using low-level features like color and texture. In this project we have implemented these two method for effective image-retrieval. Further we will now explore color and texture feature.

Color based CBIR

This is the most basic, important and widely used approach. This feature better gives perception and is insensitive to translation,rotation and scale changes. For this we use color histograms to extract features. Color space was selected as proposed in the paper now we need to define the number of bins for our histogram. Histograms are used to give an idea of the density of pixel intensities in an image. Essentially, color histogram are used to to estimate the probability density of the underlying function, or in this case, the probability of a single pixel color any C occurring in our image.

Also, we see that there is a trade-off between the number of bins for histogram. Selecting too few bins, will have less components and unable to disambiguate between images with substantially different color distributions. The color histogram is independent of the semantics of the image as this only considers the color distribution not the object present in it.

For comparing purposes we defined both global color and a local global color feature for an image.

Local histograms represents color-based segmentation approach which provide a more flexible representation and hence more powerful queries can be made. As such representation of images must be closely related to human visual perception, as this will show which part of image is compared to corresponding part and this is exactly how a human perceive the similarity between two images. So for comparing results we are considering for both global and local features.

For global color feature we used following steps:

Step1: convert image to HSV color space

Step2: count each feature value for image.

Step3: calculate Euclidean distance between the images to judge similarity between the images using formula:

$$D = \sum_{i=1}^{n} (A_i - B_i)^2.$$

For local feature we carried out as, dividing the image in nxn blocks(we used n=3) , where each block we carry out calculations same as in global color feature. Both were compared to see results.

Texture based CBIR

Texture refers to visual patterns with properties of homogeneity that do not result from the presence of only a single color such as clouds and water. Texture features typically consist of contrast, uniformity, coarseness, and density.[5] There are two fundamental classes of surface

descriptors, to be specific, statistical model-based and transform-based. The former one explores the grey-level spatial dependence of textures and then extracts some statistical features as texture representation. One example of this group is co-occurrence matrix representation[5] which we have used in this project. For this we determine horizontal and vertical space domain and direction and distance d is determined here directions used are: 0°, 45°, 90°, 135° and distance used is 1. The texture feature is extracted in 5 steps as discussed in the paper:

Step1: first color image is converted to grayscale using formulae:

$$Y = 0.29 \times R + 0.587 \times G + 0.114 \times B$$

Where Y will give the required gray-scale.

Step2: Feature value calculation: we calculate four co-occurrence matrices in four directions. The parameters which are here taken into account are moment of inertia, Angular Second Moment, Correlation and Homogenity[6].

1. MOMENT OF INERTIA(CONTRAST)

$$I = \sum_i \sum_j (i-j)^2 P(i,j)$$

Contrast is also seen as Moment of inertia, the value for it will be large for images which have a large amount of local spatial variation in gray levels and a smaller value for images with spatially uniform gray level distributions.

2. ASM(Angular Second Moment):

$$E = \sum_i \sum_j [(P(i,j))]^2$$

ASM = sqrt(E)

Energy is the measure of gray distribution uniformity of image. Here ASM is sqr root for the energy. So here the coarser the texture is, the more ASM value.

3. CORRELATION :

$$C = (d,\theta) = \frac{\sum_{i,j}(i-\mu_x)(j-\mu_y)P(i,j)}{\sigma_x \sigma_y}$$

Correlation is used to measure the degree of similarity of the blocks or elements of the image

4. HOMOGENITY :

$$H_o = \sum_i \sum_j \frac{P(i,j)}{1+|i-j|}$$

Homogeneity feature returns a value that measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal.

Then we calculate means and standard deviation for all these parameters treated as a component of the feature vector.

Step3: For each component in the feature vector we calculate normalised form of it by subtracting it by mean and dividing by standard deviation

$$h^{i,j'} = \frac{h^{i,j} - m_j}{\sigma_j}$$

Where h^{ij} is the feature component, m_j is mean and σ_j is standard deviation.

Step4: After calculating the feature vector for each image, then similarity is calculated using euclidean distance.
The results for this is discussed in the result section.

III. Implementation details

The project is implemented using python 3.6 and opencv libraries. The image dataset is selected from here[3]. These images will be stored in Image database having image information and feature vector, where the query will be made and resultant images will be retrieved, each image is in jpg format.There were three databases made. One having only global histograms for all images, second having local histograms for each image and third contain texture feature for all images.

For implementation to obtain a color histograms we do this in two ways: global feature and local feature and for texture feature were calculated for each image.For local-color feature we have divided the image into 3x3 parts making nine parts of the image and we calculate histogram for each part and that is saved in the database. When a query image is given all 9 histograms are calculated for query image and the comparisons are made to corresponding parts of all the images present in the database and resultant images are retrieved.

For Texture feature , we calculate moment of inertia, Angular Second Moment, Correlation and Homogenity for all images in all four directions. So, we get four co-occurrence for each image. Then we calculate four parameters for each matrix, and then we normalise it for parameters in different directions. So finally we get 16 values for one image in a vector format which is saved in the database.

Design issues:

1. The whole directory of images used for experiments is big (9907), thus it is taking a lot of time for insertion and retrieving results(approx 30 minutes.).
2. Texture feature is not performing well.
3. Also the database is limited in images and does not contain all sort of images, so if an image is given which does not have similar images in database it will show other images which has lower euclidean value comparatively but may be large in general.
4. Setting up threshold for euclidean distance is must here, but not able to judge what because of the issue 3 listed above.

Problems:

1. While saving the 2-D list(for histogram) to the sql table so that we can iterate it later and then compare various image features. So then we first change it to 1d array and then save it as list.
2. Here firstly while creating histograms, because of big length it was not saving in the sql table. So it was solved by changing datatype of columns to TEXT.
3. While showing results, initially separate images were shown, so finally we used PIL to combine various images according to each image height and weight and were combined.
4. There was a conflict in deciding what weightage has to be given to each feature(histogram/feature). As in paper it was mentioned that it took time for them and lot of experimentation to decide the weightage

Usage of program:

First of all the tables are to be created in mysql before inserting the directory of images and running the following programs.

So run file **tablecreation.sql** by clicking on it this will open the MYSQL workbench and execute the script, this will create all the necessary tables.

There are total three programs:

1. Program1(**Histogramcalculation.py**) (approx 26minutes for dataset provided in the link and bitbucket)

We pass directory of images to must be the format /*.jpg for jpg format image.

To run

$python Histogramcalculation.py <directoryname>/*.jpg

Result

This will insert the images information in the sql tables. One table will have a global histogram values and other table will have local features for color histograms for all parts, with names of the images.

2. Program 2 (**Texturecalculation.py**)

We pass directory of images to must be the format /*.jpg for jpg format image.

To run

$python Texturecalculation.py <directoryname>/*.jpg

Result

This will insert the images information in the sql tables. One table will have texture features informations with names of the images.

3. Program 3 (**finalProject.py**)

To run

$python Histogramcalculation.py <Query_Image_Name>.jpg

Result

This will show the four results: first window will show result for global color histograms, second window will show for local color histograms, third will be for texture histogram and finally last window will show the results for combinations of local features and texture features combined

IV. Results and discussion

A. Results

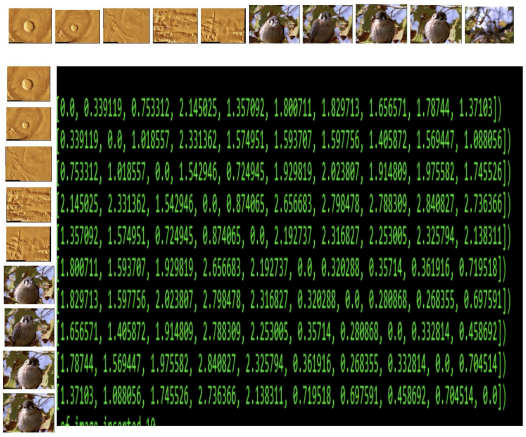
I . The first experiment we did was to test goodness of the global histogram feature. Here we show distance between a set of 10 images were taken to see the distance and we constructed a matrix format fig[2]:

Fig[2]

Fig[2] shows the result of 10 images of sunset and red car. The matrix between in green shows the euclidean distance showing the similarity between the different images.

We observe that red car had a small value of distance with other car images and have karger distance with sunset images showing that a red car image is similar to other red cars than sunset image and so sunset images with sunsets, which quite proof the good performance of the histogram. The diagonals shows similarity for same image which is zero here as shown.

II. The second experiment we did was to test goodness of the texture feature extraction. Here we show distance between a set of 10 images were taken to see the distance and we constructed a matrix format fig[3]:



Fig[3]

Fig[2] shows the result of 10 images consisting of planet surface and bird. The matrix between in green shows the euclidean distance showing the similarity between the different images using texture feature. We observe that bird images had a small value of distance with other bird images than planet surface images and so happends with surface images as well , which quite proof the good performance of the texture feature. The diagonals shows similarity for same image which is zero here as shown.

III. This experiment was performed to see which of the two global color histogram and local-color histogram performs better with query image is an elephant[fig4] and the result is shown in fig[5] for global histograms and fig[6] shows local histogram.

Fig[4]

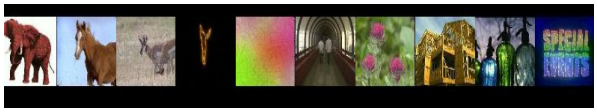
Fig[5]

Fig[6]

This shows that most of the part of the image is white as background, this global feature is considering that and showing results whose background is white. Whereas in local-feature we see that it is checking with each corresponding block for comparison. Here as each block will have some of the part of elephant which is brown in color, thus this result from local-color histogram make sense.

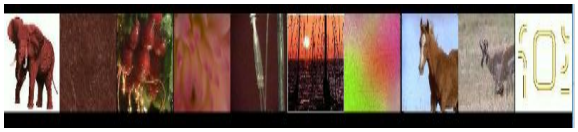
IV This experiment was performed to see the results when different weightage was given to different features(color-histograms or texture feature) to the combined program3 having both the local color feature and texture feature.

a) Result1 : it contains query image of a elephant and[fig4] this result fig[7] with weight given to texture is 0.8 and color histogram as 0.2



Fig[7]

b) Result 2 : it contains query image of a elephant and[fig4] this result fig[9] with weight given to texture is 0.5 and color histogram as 0.5



Fig[9]

Here we observe that, result contains mixture of both the features, here as query image is a brown elephant, its texture matches with horse and deer thus giving precision at 8 and 9 and because of color we are getting brown images as each block will have some part of brown, thus though results are not so accurate as in the database there are not much elephant's photos but still showing manageable results of animals with similar colored.

C. here again the experiment was performed with other query image [fig10] and the texture and color histograms were given 0.5 weightage each we see that we got some good results.
Fig[11].

fig[10]

Fig[11]

Here we see that while performing results for number of times, with other query images we found that giving equal weightage to each feature yield good result that make sense and close to human perspective.

B. Discussion

To measure the effectiveness of the CBIR is on the basis of performance of the goodness of feature extraction and similarity between the query image and retrieved ones. The two measure we used are not solely based on not just the effective image retrieval but also stability of results. They are: Precision Precision: precision in CBIR will be the fraction of relevant instances among the retrieved instances.
For testing of the precision and recall we have synthetically made a set of 25 images having five categories: . Now running both features separately and together provide the following results.

Looking at fig[11] we see that precision is at close to 1 and 3,4 as the query image contains leafs and a butterfly, the 1st image we obtained in the resultant is the same query image we got. Whereas for 3rd image contains a house covered with trees so having similar texture as query image of leaves around and so is the color is visually green. Also, 4th image is a toad with leaves around this is somehow similar in human perception.

Limitations

Color histograms have high sensitivity to noisy interference such as lighting intensity changes and quantization errors. High dimensionality (bins) color histograms are also another issue. Some color histogram feature spaces often occupy more than one hundred

dimensions.We do not yet have a universally acceptable algorithmic means of characterizing human vision, more specifically in the context of image understanding. Hence it is not surprising to see continuing efforts towards it, either building up on prior work or exploring novel directions.

References:

1. https://en.wikipedia.org/wiki/Content-based_image_retrieval
2. Jun.Y, Zhenbo.L, Lu.L , Zetian.F 2011 Content-based image retrieval using color and texture fused features DOI- https://www.sciencedirect.com/science/article/pii/S0895717710005352
3. Data link: http://imagedatabase.cs.washington.edu/groundtruth/
4. https://dsp.stackexchange.com/questions/2687/why-do-we-use-the-hsv-colour-space-so-often-in-vision-and-image-processing
5. Sridhar,Gowri, 2012, Color and Texture Based Image Retrieval DOI:http://scientific-journals.org/journalofsystemsandsoftware/archive/vol2no1/vol2no1_1.pdf
6. N.Puviarasan1, Dr.R.Bhavani2, A.Vasanthi3 2014 Image Retrieval Using Combination of Texture and Shape Features DOI:https://pdfs.semanticscholar.org/ddaa/6e0b3d228d5389d060d7fefffaaa0db98864.pdf
7. Bongani Malinga, Daniela Raicu, Jacob Furst , Local vs. Global Histogram-Based Color Image Clustering DOI: http://facweb.cs.depaul.edu/research/techreports/tr06-010.pdf

Source Codes:

Code1: Hsitogramcalculation

```
import cv2
import glob
import numpy as np
from PIL import Image
import mysql.connector # Need to import this in order to perform the SQL operations
import time

conn = mysql.connector.connect(host= "localhost",
            user="root",        ### Id and password for mysql database
            passwd="root",
            db="imagefeatures") ### Name of th database
x = conn.cursor()

import sys
```

```
infile = sys.argv[1]

def color_extraction_multiple_blocks(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    rows,cols = img.shape[:2]
    windowsize_r = int(cols/3)
    windowsize_c = int(rows/3)
    hstgm1 = []


# Crop out the window and calculate the histogram
    for r in range(0,img.shape[1] - windowsize_r, windowsize_r):
        for c in range(0,img.shape[0] - windowsize_c, windowsize_c):
            Mask = np.zeros(img.shape[:2], dtype = "uint8")
            cv2.rectangle(Mask, (r, c), (r+windowsize_r, c+windowsize_c), 255, -1)
            hist = calculate_histogram(img,Mask)

            hstgm1.extend(hist)
    return hstgm1

def calculate_histogram(img,mask):
    hstgm = cv2.calcHist([img], [0, 1,2],mask,[16,32,1],[0, 180, 0, 256,0,256])
    hstgm =  cv2.normalize(hstgm,hstgm)
    return hstgm.flatten()

def insert_feature_nultiple_blocks(filename,histo): # inserting the histogram of an image along with its
name into Mysql
    try:
        histo = str(histo)
        x.execute("""INSERT INTO blockhistogram VALUES (%s,%s)""",(filename,histo))


    except(mysql.connector.Error, mysql.connector.Warning) as e:
        print(e)
        conn.rollback()
    conn.commit()

def color_extraction(img): #Calculating the histogram of the full image

    img =cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

```
    hstm = cv2.calcHist([img], [0, 1,2],None,[16,32,1],[0, 180, 0, 256,0,256])
    hstgm =  cv2.normalize(hstm,hstm)
    return hstgm.flatten()


def insert_feature(filename,features): # inserting the histogram of an image along with its name into
Mysql
    try:
        features = str(features)
        x.execute("""INSERT INTO fullhistogram VALUES (%s,%s)""",(filename,features))


    except(mysql.connector.Error, mysql.connector.Warning) as e:
        print(e)
        conn.rollback()
    conn.commit()



# reading all the images from the specified directory
print("inserting data into sql started")
starttime = time.time()
count = 0
for filename in glob.glob(infile): #assuming jpg
    im=Image.open(filename)
    img = cv2.imread(filename)
    histo_feature = []
    histo_feature.extend(color_extraction(img))

    insert_feature(filename,histo_feature)


    histo = color_extraction_multiple_blocks(img)

    insert_feature_nultiple_blocks(filename,histo)


    im.close()
    count = count + 1

conn.close()
end_time = time.time()
print("total number of feature of image extracted and saved in Mysql",count)
```

```
print("total time taken to insert %s images is $s",count,end_time-starttime)
```

Code2: Texturecalculation

```
####This is the code to extract the texture feature from the image

import cv2
import numpy as np
from PIL import Image
import mysql.connector
import glob
from skimage import feature
from scipy.spatial import distance
import time
import sys
infile = sys.argv[1]

conn = mysql.connector.connect(host= "localhost",
                user="root",
                passwd="root",
                db="imagefeatures")
x = conn.cursor()

def texture_calculator(img):
    img =cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # converting the image into Gray scale to extract the testure feature
    comatrix = feature.greycomatrix(img,[1],[0,np.pi/4,np.pi/2,3*np.pi/4]) # Calculating the co-occurance matrix in the direction 0,45,90,135 degree direction
    ContrastStats = np.reshape(feature.greycoprops(comatrix, 'contrast'),4) # calculating the contrast of the image
    CorrelationtStats = np.reshape(feature.greycoprops(comatrix, 'correlation'),4) # calculating the correlation of the image
    HomogeneityStats = np.reshape(feature.greycoprops(comatrix, 'homogeneity'),4) # calculating the homogeneity of image
    ASMStats = np.reshape(feature.greycoprops(comatrix, 'ASM'),4)# Calculating the ASM of the image

    texture_feature = []


    normalized_contrast = [(i-(np.mean(ContrastStats)))/(np.std(ContrastStats)) for i in ContrastStats] #To normalize - take the value , substract the mean from it and then divide it by its standard deviation

#    normalized_contrast = np.asarray(normalized_contrast)
```

```
    texture_feature.extend(normalized_contrast)


    normalised_corr = [(i-(np.mean(CorrelationtStats)))/(np.std(CorrelationtStats)) for i in
CorrelationtStats] #To normalize - take the value , substract the mean from it and then divide it by its
standard deviation
    texture_feature.extend(normalised_corr)


    normalized_homo = [(i-(np.mean(HomogeneityStats)))/(np.std(HomogeneityStats)) for i in
HomogeneityStats]#To normalize - take the value , substract the mean from it and then divide it by its
standard deviation

#    normalized_homo = np.asarray(normalized_homo)
    texture_feature.extend(normalized_homo)

    normalized_asm = [(i-(np.mean(ASMStats)))/(np.std(ASMStats)) for i in ASMStats]#To normalize -
take the value , substract the mean from it and then divide it by its standard deviation

#    normalized_asm = np.asarray(normalized_asm)
    texture_feature.extend(normalized_asm)
    return texture_feature

def insert_feature(filename,texture_feature): # inserting the texture feature of an image along with its
name into Mysql
    try:
        texture_feature = str(texture_feature)
        x.execute("""INSERT INTO texture VALUES (%s,%s)""",(filename,texture_feature))


    except(mysql.connector.Error, mysql.connector.Warning) as e:
        print(e)
        conn.rollback()
    conn.commit()
count = 0
start_time = time.time()
for filename in glob.glob(infile): #assuming jpg
    im=Image.open(filename)
    img = cv2.imread(filename)
    texture = texture_calculator(img)
    insert_feature(filename,texture)
```

```
    count = count+1
    im.close()

conn.close()
end_time = time.time()
print("Total number of image inserted",count)
print("Total time taken for inserttion" ,start_time-end_time)
```

Code3:FinalProject

```
import cv2
import numpy as np
from PIL import Image
import mysql.connector
from skimage import feature
from scipy.spatial import distance
import math
import operator
import PIL
import sys
import time
conn = mysql.connector.connect(host= "localhost",
                user="root",
                passwd="root",
                db="imagefeatures")
x = conn.cursor()
def single_histogram(img): #Calculating the histogram of the full image
    img =cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    hstm = cv2.calcHist([img], [0, 1,2],None,[16,32,1],[0, 180, 0, 256,0,256])
    hstgm =  cv2.normalize(hstm,hstm)
    return hstgm.flatten()

def texture_calculator(img):
    img =cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    comatrix = feature.greycomatrix(img,[1],[0,np.pi/4,np.pi/2,3*np.pi/4])
    ContrastStats = np.reshape(feature.greycoprops(comatrix, 'contrast'),4)
    CorrelationtStats = np.reshape(feature.greycoprops(comatrix, 'correlation'),4)
    HomogeneityStats = np.reshape(feature.greycoprops(comatrix, 'homogeneity'),4)
    ASMStats = np.reshape(feature.greycoprops(comatrix, 'ASM'),4)

    texture_feature = []
```

```
    normalized_contrast = [(i-(np.mean(ContrastStats)))/(np.std(ContrastStats)) for i in ContrastStats]

#    normalized_contrast = np.asarray(normalized_contrast)



    texture_feature.extend(normalized_contrast)


    normalised_corr = [(i-(np.mean(CorrelationtStats)))/(np.std(CorrelationtStats)) for i in
CorrelationtStats]
    texture_feature.extend(normalised_corr)


    normalized_homo = [(i-(np.mean(HomogeneityStats)))/(np.std(HomogeneityStats)) for i in
HomogeneityStats]

#    normalized_homo = np.asarray(normalized_homo)
    texture_feature.extend(normalized_homo)

    normalized_asm = [(i-(np.mean(ASMStats)))/(np.std(ASMStats)) for i in ASMStats]

#    normalized_asm = np.asarray(normalized_asm)
    texture_feature.extend(normalized_asm)
    return texture_feature


def color_extraction(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    rows,cols = img.shape[:2]
    windowsize_r = int(cols/3)
    windowsize_c = int(rows/3)
    hstgm1 = []


# Crop out the window and calculate the histogram
    for r in range(0,img.shape[1] - windowsize_r, windowsize_r):
        for c in range(0,img.shape[0] - windowsize_c, windowsize_c):
            Mask = np.zeros(img.shape[:2], dtype = "uint8")
            cv2.rectangle(Mask, (r, c), (r+windowsize_r, c+windowsize_c), 255, -1)
            hist = calculate_histogram(img,Mask)
```

```
            hstgm1.extend(hist)
    return hstgm1

def calculate_histogram(img,mask):
    hstgm = cv2.calcHist([img], [0, 1,2],mask,[16,32,1],[0, 180, 0, 256,0,256])
    hstgm =  cv2.normalize(hstgm,hstgm)
    return hstgm.flatten()




def read_single_histogram_features_from_Index():
    conn = mysql.connector.connect(host= "localhost",
                user="root",
                passwd="root",
                db="imagefeatures")
    x = conn.cursor()
    sql="select filename,Histogram from fullhistogram;"
    try:
        x.execute(sql)
        data = x.fetchall()
    except(mysql.connector.Error, mysql.connector.Warning) as e:
        print(e)
        conn.rollback()
    conn.close()
    return data

def read_multiple_histogram_features_from_Index():
    conn = mysql.connector.connect(host= "localhost",
                user="root",
                passwd="root",
                db="imagefeatures")
    x = conn.cursor()
    sql="select filename,Histogram from blockhistogram;"
    try:
        x.execute(sql)
        data = x.fetchall()
    except(mysql.connector.Error, mysql.connector.Warning) as e:
        print(e)
        conn.rollback()
    conn.close()
    return data
```

```
def read_texture_features_from_Index():
    conn = mysql.connector.connect(host= "localhost",
                user="root",
                passwd="root",
                db="imagefeatures")
    x = conn.cursor()
    sql="select filename,texture_feature from texture;"
    try:
        x.execute(sql)
        data = x.fetchall()
    except(mysql.connector.Error, mysql.connector.Warning) as e:
        print(e)
        conn.rollback()
    conn.close()
    return data



def read_color_texture_features_from_Index():
    conn = mysql.connector.connect(host= "localhost",
                user="root",
                passwd="root",
                db="imagefeatures")
    x = conn.cursor()
    sql="select t.filename,b.Histogram,t.texture_feature from  blockhistogram as b,texture as t where
t.filename = b.filename;"
    try:
        x.execute(sql)
        data = x.fetchall()
    except(mysql.connector.Error, mysql.connector.Warning) as e:
        print(e)
        conn.rollback()
    conn.close()
    return data


def calc_distance(features,color_histo_query): #calculating the eculidean distance of the fetaure vectors
    color_dist = math.sqrt(sum([(x-y)**2 for x,y in zip(features, color_histo_query)]))

    return color_dist
```

```
def
calc_distance_text_color(color_multiple_histo_qurey,color_features,texture_feature_query,text_feat):#cal
culating the distance of texture and color histogram

    color_dist = math.sqrt(sum([(x-y)**2 for x,y in zip(color_multiple_histo_qurey, color_features)]))

    text_dist  = math.sqrt(sum([(x-y)**2 for x,y in zip(texture_feature_query, text_feat)]))

    return (0.5 *color_dist + 0.5*text_dist )



qurey_image = cv2.imread(sys.argv[1]) ### Accepting the qurey image




color_histo_query = single_histogram(qurey_image)
texture_feature_query = texture_calculator(qurey_image)
##################### calculating the histogram of qurey image and compare it with all the
##################### images in database and display the best 10 matches
start_time = time.time()
histo_data_feature = read_single_histogram_features_from_Index()

single_histo_dist = {}
for fid, feature1 in histo_data_feature:
#    print(type(feature1))
    features = [float(x) for x in feature1.strip('[]').split(',')]

    dist = calc_distance(features ,color_histo_query)

    single_histo_dist[fid] = dist
single_histo_dist = sorted(single_histo_dist.items(), key=operator.itemgetter(1))

sorted_image = single_histo_dist[:10]
#print("\n\n image for single histogram",single_histo_dist[:10])
image_address = []

for i in sorted_image:
    img_address, img_distance = i
    image_address.append(img_address)
```

```
imgs = [PIL.Image.open(i) for i  in image_address]
min_shape = sorted([(np.sum(i.size),i.size) for i  in imgs])[0][1]
imgs_comb = np.hstack((np.asarray(i.resize(min_shape))for i in imgs))
imgs_comb = PIL.Image.fromarray(imgs_comb)
imgs_comb.save('trial.jpg')


image = PIL.Image.open('trial.jpg')

image.show()
end_time = time.time()
#print("total time take for retirvel based on single histogram",end_time - start_time)


###############################################################################
###############################################################################
###############################################################################
#################### calculating the block histogram of qurey image and compare it with all the
#################### images in database and display the best 10 matches

start_time = time.time()
color_multiple_histo_qurey = color_extraction(qurey_image)

histo_multiple_data_feature = read_multiple_histogram_features_from_Index()
multiple_histo_dist = {}
count = 1
for fid, feature1 in histo_multiple_data_feature:
#    print(type(feature1))
    features = [float(x) for x in feature1.strip('[]').split(',')]

    dist = calc_distance(features ,color_multiple_histo_qurey)

    multiple_histo_dist[fid] = dist
    count = count +1
#    print(count)
multiple_histo_dist = sorted(multiple_histo_dist.items(), key=operator.itemgetter(1))

soretd_image_multiple = multiple_histo_dist[:10]
#print("\n\n\image for multiple histogram",multiple_histo_dist[:10])

image_address_multiple = []

for i in soretd_image_multiple:
```

```
    img_address_multiple, img_distance = i
    image_address_multiple.append(img_address_multiple)

imgs = [PIL.Image.open(i) for i  in image_address_multiple]
min_shape = sorted([(np.sum(i.size),i.size) for i  in imgs])[0][1]
imgs_comb = np.hstack((np.asarray(i.resize(min_shape))for i in imgs))
imgs_comb = PIL.Image.fromarray(imgs_comb)
imgs_comb.save('trial1.jpg')
image = PIL.Image.open('trial1.jpg')
image.show()

end_time = time.time()
print("total time take for retirvel based on block histogram",end_time - start_time)



##########################################################################################
##########################################################################################
##########################################################################################
##########################################################################################
#################### calculating the texture of qurey image and compare it with all the
#################### images in database and display the best 10 matches
start_time = time.time()
texture_feature = read_texture_features_from_Index()
texture_dist = {}
for fid, feature1 in texture_feature:
#    print(type(feature1))
    features = [float(x) for x in feature1.strip('[]').split(',')]

    dist = calc_distance(features ,texture_feature_query)

    texture_dist[fid] = dist
texture_dist = sorted(texture_dist.items(), key=operator.itemgetter(1))
soretd_image_texture = texture_dist[:10]
print("\n\n\image for texture histogram",texture_dist[:10])

image_address_texture = []

for i in soretd_image_texture:
    img_address_texture, img_distance = i
    image_address_texture.append(img_address_texture)

imgs = [PIL.Image.open(i) for i  in image_address_texture]
```

```
min_shape = sorted([(np.sum(i.size),i.size) for i  in imgs])[0][1]
imgs_comb = np.hstack((np.asarray(i.resize(min_shape))for i in imgs))
imgs_comb = PIL.Image.fromarray(imgs_comb)
imgs_comb.save('trial2.jpg')
image = PIL.Image.open('trial2.jpg')
image.show()
end_time = time.time()
print("total time take for retirvel based on texture",end_time - start_time)
##########################################################################################
##########################################################################################
##########################################################################################
#################### calculating the histogram  and texture of qurey image and compare it with all the
#################### images in database and display the best 10 matches
start_time = time.time()
count = 1
color_texture_feature_retirved = read_color_texture_features_from_Index()
color_texture_feature = {}
for fid,color_hist,texture_feats in color_texture_feature_retirved:
#    print(fid)
    color_features = [float(x) for x in color_hist.strip('[]').split(',')]

    text_feat = [float(x) for x in texture_feats.strip('[]').split(',')]

    dist =
calc_distance_text_color(color_multiple_histo_qurey,color_features,texture_feature_query,text_feat)

    color_texture_feature[fid] = dist
    count = count+1
#    print(count)

color_texture_feature = sorted(color_texture_feature.items(), key=operator.itemgetter(1))
soretd_image_texture_col = color_texture_feature[:10]
#print("\n\n\image for color feature texture histogram",color_texture_feature[:10])


image_address_color_texture = []

for i in soretd_image_texture_col:
    img_address_texture_col, img_distance = i
    image_address_color_texture.append(img_address_texture_col)

imgs = [PIL.Image.open(i) for i  in image_address_color_texture]
min_shape = sorted([(np.sum(i.size),i.size) for i  in imgs])[0][1]
```

```
imgs_comb = np.hstack((np.asarray(i.resize(min_shape))for i in imgs))
imgs_comb = PIL.Image.fromarray(imgs_comb)
imgs_comb.save('trial3.jpg')
image = PIL.Image.open('trial3.jpg')
image.show()
end_time = time.time()
print("total time take for retirvel based on texture and color histogram",end_time - start_time)


#=============================================================================
====
# for i in single_histo_dist[:10]:
#     img_name = i.strip().split(",")
#     print(i)
#=============================================================================
====

conn.close()
```