Colored Image Retrieval based on Most used Colors

Sarmad O. Abter
Department of Computer Science
University of Baghdad
Baghdad, Iraq

Dr. Nada A.Z Abdullah
Department of Computer Science
University of Baghdad
Baghdad, Iraq

Abstract—The Fast Development of the image capturing in digital form leads to the availability of large databases of images. The manipulation and management of images within these databases depend mainly on the user interface and the search algorithm used to search these huge databases for images, there are two search methods for searching within image databases: Text-Based and Content-Based. In this paper, we present a method for content-based image retrieval based on most used colors to extract image features. A preprocessing is applied to enhance the extracted features, which are smoothing, quantization and edge detection. Color quantization is applied using RGB (Red, Green, and Blue) Color Space to reduce the range of colors in the image and then extract the most used color from the image. In this approach, Color distance is applied using HSV (Hue, Saturation, Value) color space for comparing a query image with database images because it is the closest color space to the human perspective of colors. This approach provides accurate, efficient, less complex retrieval system.

Keywords—Most used colors feature; color histogram; content-based image retrieval (CBIR); contour analysis; HSV color space

I. INTRODUCTION

This paper demonstrates the ability of computer system of retrieval of the images based on the color similarity, the paper first includes introduction to the theory and history of the Content-Based Image Retrieval (CBIR) then browse through some of the similar works with this work and then demonstrates the proposed system in details finally the results are shown and discussed briefly.

We chose this domain of work because of the need in some areas for an application that is able to retrieve images based on their visual content.

The development of the computer network and image capturing and processing devices and computer-aided image generation applications that produce images within computers led to the invention and creation of large image databases that have large numbers of images with different classes of visual information contained within it and for that is one of the main reasons that the researchers [11] gave a great attention to find a way to search within these databases and retrieve images accurately and within an acceptable amount of time this method of search is called content-based image retrieval (CBIR) or query by image content (QBIC).

There exist a lot of application domains in which CBIR is very important [1]. Examples of some areas are; Weather forecasting, Military, GIS systems, Criminal Investigation, Bio-Medical Imaging, Scientific database, Surveillance systems, Remote Sensing (Satellites). In many areas of commerce, government, academia, and hospitals, large collections [13] of digital images are being created [12]. Many of these collections are the product of digitizing existing collections of analog [3] photographs, diagrams, drawings, paintings, and prints. Usually, the only way of searching these collections was by keyword indexing, or simply by browsing [5]. Digital images databases, however, open the way to content-based searching. There are various technical aspects of current content-based image retrieval systems and a number of other overviews on image database systems, image retrieval, or multimedia information systems have been published [2].

The methods to search within large image databases before the invention of this [8] method is called Text-Based Image Retrieval (TBIR) in which the metadata or tags about the contents of the images are added manually to the images [14]. This operation takes a lot of time and effort especially for a large number of images and does not provide an accurate description of the visual content of the image which may lead to inaccurate search results. This image search method depends on a comparison between the search term and the tags or names of the image files without any noticing to the content of the image [6].

The Content-Based Image Retrieval has come to avoid these challenges in the Text-Based Image Retrieval (TBIR) in which the visual features of the images of the database are extracted and formed a feature vector that will be stored in the feature database to be compared later with the query image feature vector that is extracted from it automatically.

Then the feature vector of the query image will be compared with the feature vector of all image of the image database that is stored in the feature database to get the most similar and relevant images from the database and make sure that the results are accurate as possible.

II. RELATED WORKS

Gauri Deshpande et al. [11] used two low-level feature which are Color and Texture for Color Feature Extraction the RGB color space was converted into HSV space and YCbCr space and for the Texture Feature the co-occurrence matrix was used and the low level that would be used depends on the application for natural images the color feature gave the best results while for the textured images the co-occurrence matrix would be suitable.

Sandhya R. Shinde et al. [9] used the color feature that was extracted from the image and applied the data preprocessing on it then the machine learning classifiers were applied to these
features to classify the images. The accuracy of the classification was measured using two criteria: color spaces and image size.

Ashutosh Gupta et al. [4] increased the efficiency and accuracy of the system by hybridizing the three main techniques of Content-Based Image Retrieval (Color, Shape, Texture) using color histogram and Color Correlogram for color feature extraction, BDIP (Block Difference of Inverse Probabilities) and BVLC (Block Variation of Local Correlation) which were block-based techniques used for shape and texture features extraction, respectively.

Abdolreza Rashno et al. [7] proposed a novel and new CBIR scheme based on the ant colony optimization (ACO) and color feature and the wavelet transformation is used for texture feature extraction.

Rajeev Srivastava et al. [10] proposed a method to classify the query image by class analyzes and eliminating the irrelevant classes that affected greatly on the results of the retrieval of the images from the database.

III. PROPOSED SYSTEM

A Content-Based Image Retrieval (CBIR) is proposed that depends mainly on the color of the object in the retrieval process, the system is able to retrieve objects with similar colors within the same class or from multiple different classes.

Our proposed system consists of two phases of operation:

The first one is the training phase: In this phase, the color feature will be extracted from all the images within the selected classes from the image Dataset and store them as a feature vector into the feature Database.

The second phase is the training phase in which the color feature of the query image is extracted and stored as a feature vector.

Then the Retrieval of the images is done by comparing the feature vector of the query image and the feature vectors that are stored in the Feature Database using the Euclidean Distance (ED) to check the similarity of the colors.

Fig. 1 shows the general diagram of the proposed system.

Before the extraction of a color feature from images preprocessing is applied to the images first, the preprocessing in this system contains the following steps:

1) Smoothing Filter

This step is used to remove the noise and blur from the image. The noise in the image represents undesirable information in the image, the noise can be an unwanted line or small dots. Noise produces undesirable effects such as artifacts, unrealistic edges, unseen lines, corners, blurred objects and disturbs background scenes. To reduce these undesirable effects, the Gaussian smoothing filter is used:

\[ G(x, y) = \frac{1}{2\pi \sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \]  

Fig. 2. Color quantization result.

(a) Original image  
(b) Image after quantization

2) Color Quantization

The Color Quantization step is an important step in the preprocessing stage because it reduces the number of color ranges from thousands of colors into few hundred colors by merging the most similar colors into single color without altering the general shape of the colored image or producing any distortion in the image contents.

The Color Quantization also have the advantage that the reduced color image would be easier to process in the similarity matching stage because the number of calculations required into match two images is reduced because the number of colors is reduced the result of applying the color quantization is illustrated in Fig. 2, the figure illustrates the image before and after quantization.
Quantization algorithm in this work depends on the image histogram, since the histogram can show the colors that are occurred most frequently, and these colors can be used to perform quantization. The method which is used in the implementation is divided into two stages of processing the first step is to calculate the color palette of the original image using color. The second stage of the quantization process is to take the color histogram of the original image that is calculated in the first stage, and divided the histogram into 256 regions, the division is decided by the following equation:

\[ NPR = \frac{NP}{NR} \]  

Where,
- NPR: number of pixels in region
- NP: number of pixels of the image
- NR: number of regions

From each region one color is selected which has maximum occurrence in the region, at the end of each region the number of pixels of the next region is calculated as in the following equations:

\[ newNP = NP - NPR \]  
\[ newNR = NR - 1 \]  
\[ newNPR = \frac{newNP}{newNR} \]  

Where,
- NPR: number of pixels in the previous region
- NP: number of pixels of the image
- newNP: number of pixels of unprocessed colors
- NR: number of regions
- newNR: number of remaining regions
- newNPR: number of pixels in next region

Color quantization algorithm illustrated in Algorithm 1.

**Algorithm (1): Color Quantization**

**Objectives:** Reduce the Colors in the Image.

**Input:** bmp //Color image

**Region_no */ //Number of colors we want to minimize

**The original number of colors of the image to it.**

**Output:** Colored Image with reduced colors.

**Step1:** Calculate the Histogram for the colors of the image scan all the pixels of the images and store the colors of the pixel in the Table

For all rows from 0 to width of the image do

For all columns from 0 to height of the image do

Get the pixel color in the row and column location

If pixel color is in the Table then
Add 1 to the value of that color
Else
Add the color to the Table and set it to 1
End For
End For

**Step2:** Find the number of pixels in each region.

Set Sum_all ← number of pixels of the image.
Set no_of_pixels ← Sum_all / Region_no // Determine the number of pixels in each region.
Set ci ← 0 // Histogram Color Index.
Set sum1 ← Sum_all

**Step3:** Determine the colors in each region and find the max color of each region.

For i from 0 to Region_no do

Begin
Set counter ← 0 // Number of colors in each region.
Set sum ← 0 // Sum of pixels of the colors in each region.
Set max_color ← color[ci] // Set the maximum color of the region.
Set max_pixel ← pixel[ci] // Set the maximum pixel of the maximum color.
While sum < no_of_pixels do // Sum the number of pixels of each color in the histogram until the number of pixels of the region is reached
Begin
Set sum ← sum + Pixel_no
If max_pixel < pixel[ci] do
Set max_color ← color[ci]
Set max_pixel ← pixel[ci]
End
Counter + 1 // Increment the counter of colors in the region.
End
End For

Set sum1 ← sum1 – sum // Subtract the sum of pixels of the first region from the sum of all pixels in the image.
Set no_of_pixels ← sum1 / (Region_no-1) // Determine the new number of pixels in each region.
Max_array[i] ← max_color // Add the maximum color in each region to array.
No_colors[i] ← counter // Add the number of colors in each region into No_colors array.
End

**Step4:** Find the distance of each color in the histogram and the colors of the Max_array to find the nearest max color to it using equation (1).

**Step5:** Create the mapping table by adding the color and the nearest max color corresponding to it to the table.

**Step6:** Set the colors of the image according to the mapping table.

**End:**

3) **Edge Detection**

The Edge Detection is required a step-in order to detect the outline of the objects in the image to efficiently extract the colors of them later. Fig. 3 illustrates the result of applying the Contour Analysis (CA).
In most researches, the object is represented using a closed contour line that surrounds the object from all sides. In this work, the Contour Analysis method is used to determine the edges or the contours of the objects in the image. In which the algorithm starts with an initial point in the image and moves with the curves of the object to connect the related lines together.

4) Find Region of Interest (ROI)

The Region of Interest (ROI) represents the area in the image that is desired to extract it from the complete image space and extract the Visual Features from it, the resulted area represents the objects in the image which we want to identify to process them further in our system to finally extract the color feature of that area of image, Fig. 4 illustrates the Region of Interest (ROI) Extraction from the image.

The steps of finding the Region of Interest used in this work are explained in Algorithm 2:

<table>
<thead>
<tr>
<th>Algorithm (2): Find Region of interest</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> gray image with the detected objects borders // Bitmap image file</td>
</tr>
<tr>
<td>Source image // Bitmap image</td>
</tr>
<tr>
<td><strong>Output:</strong> colored image with colored objects // Bitmap image file</td>
</tr>
</tbody>
</table>

**Begin**

**Step 1:** Extract Red, Green, Blue arrays from the image
   For all rows from 0 to width of the image do
      For all columns from 0 to height of the image do
         ImageR[ row, column ] ← image[ row, column ]. R
         ImageG[ row, column ] ← image[ row, column ]. G
         ImageB[ row, column ] ← image[ row, column ]. B
      End For
   End For

**Step 2:** set the color of 5*5 borders into white color
   For all rows from 0 to width of the image do
      For all columns from 0 to height of the image do
         if ((row < 5) or (column < 5) or (row > width - 5) or (column > height - 5))
            Set image[ row, column ] ← color. White
         End if
      End for
   End for

**Step 3:** scan the rows and columns to find the first black color in row and column and save their locations
   For all rows from 0 to width do
      For all columns from 0 to height do
         If image[ row, column ] is black
            column1 ← column
         End if
      End for
   End for
   For all columns from height-5 to column1 do
      If image[ row, column ] is black
         Column2 ← column
      End if
   End for

**Step 4:** Retrieve the color of the image between column1 and column2
   For all rows from 0 to width do
      For columns from column1 to column2 do
         Get the color of the source image in the specified location
         Set the color of the image with that color
      End for
   End for

**End**

5) Feature Extraction

The feature extraction is done in this work by calculating the Histogram for the image after applying the Contour Analysis algorithm and identifying the Region of Interest (ROI). The Histogram is passed as an argument into the
Quantization Algorithm 1 to reduce the colors that will be used as a feature vector to be used in the color comparing stage of the system into eight Colors then save these eight colors into feature database as a feature vector for each image.

Algorithm (3): Feature Extraction

Input: colored image with eliminated background // BMP image file
Output: (8) most used colors in the image // Table of colors

Begin
Step1: Calculate the histogram for the image after Contour Analysis (CA)
For all rows from 0 to width of the image do
For all columns from 0 to height of the image do
If image [row, column] is in the Table then
Add 1 to the value of that color
Else
Add the color to the Table and set it to 1
End For
End For
Step2: pass the Histogram into Algorithm (1) to reduce the colors into (8) colors
Step3: save the (8) colors into feature database as a feature vector.
End;

6) Similarity Measure
The similarity measure between colors extracted from the query image and the colors stored in the feature database are done using the HSV (Hue, Saturation, Value) color Space, the similarity is done using the Euclidean Distance Equation as follows:

\[
\begin{align*}
\text{d}(h_1, h_2) &= \min\{\text{abs}(h_2 - h_1), 360 - \text{abs}(h_2 - h_1)\} / 180.0 \quad (6) \\
\text{d}(s_1, s_2) &= \text{abs}(s_1 - s_2) \quad (7) \\
\text{d}(v_1, v_2) &= \text{abs}(v_1 - v_2) / 255.0 \quad (8) \\
\text{d}(h, s, v) &= \sqrt{(\text{d}(h_1, h_2))^2 + (\text{d}(s_1, s_2))^2 + (\text{d}(v_1, v_2))^2} \quad (9)
\end{align*}
\]

IV. RESULTS AND DISCUSSION
In this paper, the system was implemented using C sharp 2015 programming language, on CPU 2.40 GHz with 6 GB RAM. The system is tested and evaluated using the CorelDB a free image data set that contains 10,800 images in 80 different groups (e.g. car, castle, bus, aviation, etc.) each class is divided into two groups 75 images as training samples and we have chosen 5 random sample images from 6 classes (cars, buses, aviation, flowers, flags, and trains) of the CorelDB to test the proposed system and the results are shown in Fig. 5, 6 and 7 while the precision and recall are shown in Table 1 for each class. Table 2 shows the comparison of the results of our system and the results of the other systems.

The precision and recall are calculated using the following equations:

\[
\begin{align*}
\text{Precision} &= \frac{\text{number of relevant images retrieved}}{\text{total number of images retrieved}} \quad (10) \\
\text{Recall} &= \frac{\text{number of relevant images retrieved}}{\text{total number of relevant images}} \quad (11)
\end{align*}
\]

As illustrated in Table 2 the method presented in this paper has better results than other similar systems.
V. CONCLUSION

In this paper, we proposed content-based color image retrieval based on the color feature. We use true color RGB histogram to reduce the colors of images into 256 colors first, and then we extract the shape using contour to find the region of interest. From the histogram of the region of interest, the most used eight colors are extracted.

The most used colors for all trained images of the dataset (CorelDB) are stored in Database. Since HSV color space is close to human visual perception, we choose HSV color space in this study for find similarity between tested images and the trained images by comparing their most used colors. During testing, the algorithm produced good results in that it was able to retrieve many relevant images.

In the future, we plan to extract the shape features in addition to color features using one of the shape extraction techniques such as invariant moments or Fourier descriptor.

The researchers which will work on similar projects have to combine multiple features and use different methods to extract the features from the image to improve the results of the system.

REFERENCES