Image Storage and Retrieval Using Color Quantization Method

Koushik Kanti Mandal[+], Md. Rafiqul Islam[+],
Mohd. Noor Md. Sap[++], Fahmida Khatun[+], Md. Parvez Reza[+]

[+] Computer Science and Engineering Discipline
Khulna University, Khulna-9208, Bangladesh
[++] Faculty of Computer Science and Information System,
University Teknologi Malaysia, Skudai,
Johor Bahru, Malaysia

Abstract: *Color is the most dominant and distinguishing visual feature for image. We propose a compact color descriptor here, which is based on the observation that a small number of colors are sufficient to characterize the color information in an image region. The image is segmented by our proposed segmentation algorithm, which uses the compact color descriptor .The result of the experiment shows that the developed segmentation algorithm produces better performances as compared with the existing segmentation algorithm. After segmentation the color features are extracted from the segmented image. Finally the color features are stored in the database using a special type of indexing method for storing and retrieval of image features.*

Keywords: Image Indexing, Image Segmentation, Color Codebook, Dominant Regions, and Hashing.

1. Introduction

Due to the rapid advance of technologies in information processing and internet communication, people now-a-days have been overwhelmed by the fast accumulation of digital information such as text, image, video, and audio around the world. This advance technology in computer and communication is pushing the existing information processing tools to their limits. The past few years have seen an overwhelming accumulation of media-rich digital data such as images, video, and audio. Several systems have been developed recently to search through image database. The use of low-level visual features to retrieve relevant information from image and video database

has drawn much research attention in recent years. Color is perhaps the most dominant and distinguishing visual feature. Using color to index and search images, color histogram [1, 2] is easy to compute, but its drawback is, it results in large feature vectors that are difficult to index and have high search and retrieval cost. In addition, spatial information is not preserved in color histogram. Several of the recently proposed color descriptors try to incorporate spatial information to varying degrees. These include the compact color moments [3, 4], binary color sets [5], color coherence vector [6], and color correlograms [7]. The feature vector dimensions of typical color descriptors are quite large. For example, the number of bins in a typical color histogram ranges from few tens to a hundreds. The high dimensionality of feature vectors result in high computational cost in distance calculation for similarity retrieval [2], and is inefficient in indexing and searching. All the pixels are considered to make calculations and define the corresponding color class labels [8]. So, the time and computational complexity is very high in case of large size images.

To overcome the difficulties of high search and retrieval cost due to high dimensionality of feature vectors the basic idea is to reduce the space of all possible colors such that the features can be efficiently organized for future search. So we propose a compact color feature descriptor in this paper. The entire RGB color space is described using a small set of color categories. A smaller set is more useful since it gives a coarser description of the color of a region thus allowing it to remain same for some variations in imaging conditions. The proposed color feature descriptor is quite compact, and based on the observation that a small number of colors are usually sufficient to characterize the color information in an image region. Since the descriptor extracts the representative or dominant colors in a given region, we refer to it as dominant color descriptor. A fast segmentation algorithm is proposed here. And finally a special type of indexing is used for storing and retrieving image features.

The paper is organized as follows: Section 1 describes the Introduction of this paper. In section 2, the use of color for image storage and retrieval are described. The proposed segmentation and indexing techniques are described details in section 3. Section 4 represents the implementation issue, experimental result with performance of our method. Section 5 describes conclusion, limitation and future plans.

2. Use of Color for Image Query

Color is one of the most dominant and distinguishing visual features. The basic idea of using color for indexing and searching is to reduce the space of all possible colors such that the feature can be efficiently organized for future search. The color descriptor can be of varying length depending on the complexity of the homogeneous region. The resulting number of color clusters in the experiment is in the range of 3-12 [9]. The color feature is then defined as

$$f_c = \{(C_j,\ P_j)\ |\ C_j \in \{1,\ 2,\ \ldots,\ 256\},\ 0 \le P_j \le 1,$$
$$\sum_{1 \le j \le N} P_j = 1,\ \text{and}\ \ 1 \le j \le N\} \quad \ldots \quad \ldots \quad \ldots \quad (1)$$

where C_j is the index into the color codebook C, P_j is the corresponding percentage, and N is the total number of colors in the region.

3. Proposed Segmentation and Indexing Technique

Our indexing approach employs color quantization and clustering (if needed) that is carried out to extract dominant regions in images based on human perceptual colors. Fewer colors (typically 4-12 per region) can thus be used to represent region color without affecting the perceptual quality. For example, a field of yellow poppy flowers has typically two dominant colors, yellow and green. The image itself may contain more than just this flowerbed.

Algorithm 3.1: Algorithm of the Proposed Method for the Storing and Retrieval of Image

1. At first take a color image as an input.
2. Then segment the image into dominant regions using the colors in the color codebook.
3. Take the color feature (percentage values of each colors that present in the segmented image).
4. Store the extracted features in the database using the indexing technique.

5. Take the query image and extract color features similarly.
6. Compare the features of the query image with the features stored in database. If any match occurs then the query image is found, otherwise not found.

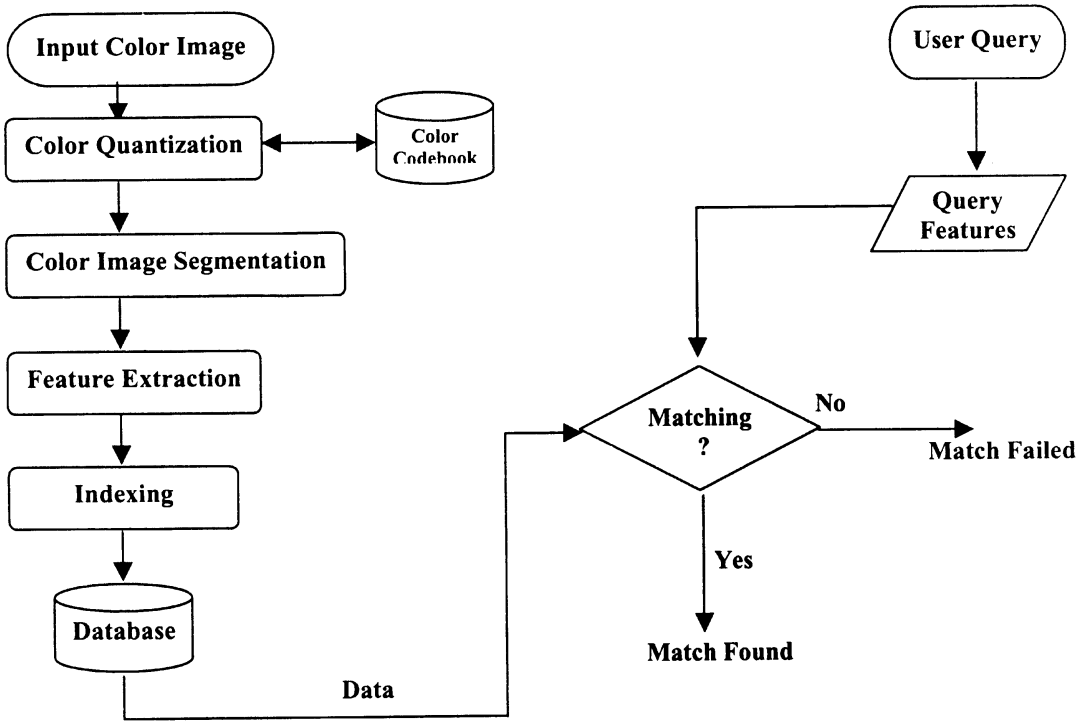

Figure 3.1: Flow Chart of Image storage and Retrieval System.

Figure 3.1 shows the flow chart of our proposed method for image storage and retrieval. At first we take a color image as input. Then quantize the input image using the color codebook and the quantized image are segmented into some regions. Then from the segmented image we calculate the percentage of each region in the entire image. Finally store only those regions' percentage values, which are greater than a threshold value in database using our proposed indexing technique. In case of searching an image user gives some query information. If these information matches with that of any image stored in database then the image is found in the database, otherwise not found. Now we see each steps of our proposed system in details.

3.1 Construction of Color Codebook

In the proposed segmentation method entire RGB (Red, Green, Blue) color space is described using a small set of color categories. A smaller set is more useful since it gives a coarser description of the color of a region thus allowing it to remain same for some variations in imaging conditions.

Our color codebook [10] is a fixed database consisting of 216 colors. Each color is represented by a triplet (Red, Green, Blue), in which red, green, and blue are three bytes that represent the basic color components. The smallest value, 0, indicates the absence of color. The largest value, 255, indicates full intensities or saturation. The triplet (0,0,0) is black, because all colors are missing, and the triplet (255,255,255) is white. The colors are chosen from 16581375(*i.e.* 255 * 255 * 255) colors by selecting RGB values at equal interval. We consider here 51 as the interval value. We take (0,0,0) as the first color and (255,255,255) as the last color. So the colors in the codebook are

Table 3.1: Tabular Representation of Color Codebook.

(R G B)	(R G B)	(R G B)	(R G B)	(R G B)	(R G B)
(0 0 0)	(0 0 51)	(0 0 102)	(0 0 153)	(0 0 204)	(0 0 255)
(0 51 0)	(0 51 51)	(0 51 102)	(0 51 153)	(0 51 204)	(0 51 255)
(0 102 0)	(0 102 51)	(0 102 102)	(0 102 153)	(0 102 204)	(0 102 255)
(0 204 0)	(0 204 51)	(0 204 102)	(0 204 153)	(0 204 204)	(0 204 255)
(0 255 0)	(0 255 51)	(0 255 102)	(0 255 153)	(0 255 204)	(0 255 255)
(51 0 0)	(51 0 51)	(51 0 102)	(51 0 153)	(51 0 204)	(51 0 255)
(102 0 0)	(102 0 51)	(102 0 102)	(102 0 153)	(102 0 204)	(102 0 255)
...	...	...	...	...	...
(51 255 0)	(51 255 51)	(51 255 102)	(51 255 153)	(51 255 204)	(51 255 255)
(102 0 0)	(102 0 51)	(102 0 102)	(102 0 153)	(102 0 204)	(102 0 255)
...	...	...	...	...	...
(102 255 0)	(102 255 51)	(102 255 102)	(102 255 153)	(102 255 204)	(102 255 255)
(153 0 0)	(153 0 51)	(153 0 102)	(153 0 153)	(153 0 204)	(153 0 255)
...	...	...	...	...	...
(153 255 0)	(153 255 51)	(153 255 102)	(153 255 153)	(153 255 204)	(153 255 255)
(204 0 0)	(204 0 51)	(204 0 102)	(204 0 153)	(204 0 204)	(204 0 255)
...	...	...	...	...	...
(204 255 0)	(204 255 51)	(204 255 102)	(204 255 153)	(204 255 204)	(204 255 255)
(255 0 0)	(255 0 51)	(255 0 102)	(255 0 153)	(255 0 204)	(255 0 255)
...	...	...	...	...	...
(255 255 0)	(255 255 51)	(255 255 102)	(255 255 153)	(255 255 204)	(255 255 255)

3.2 Color Quantization and Segmentation

The first step in color image indexing scheme is to extract color features. Inexpensive display hardware stores 8 bits per pixel, so it can display at most 256 distinct colors at a time. To display a full-color image, the computer must choose an appropriate set of representative colors and map the image into these colors. This process is called "color quantization". A good color quantization is important to the segmentation process.

The image is segmented into regions according to their perceived color *i.e.* mapping all pixels to their categories in color space, and grouping pixels belonging to same category. A color will be selected from 216 pre-defined colors in color codebook, which is very near to image pixel color, and it will be stored as new color for that pixel in the image. Using P the image pixel's color value and C the corresponding color codebook value, Color distance, C_{dis} is calculated using Euclidean distance formula, which is specified in equation 2. For a pixel we calculate the color distance with all the colors in the color codebook. Then sort these distance values in ascending order and select the color for which the distance is minimum, as the new color for that pixel. For segmentation we do not calculate the minimum distance for all the pixels in an image. We start from the top left pixel of the image and approach in left to right and top to bottom manner. We store the RGB components of the pixels of the image after certain interval T_d. We define T_d with respect to the spatial coordinates by our experiment where T_d is a small integer. If T_d is large then the segmentation time reduces but the quality becomes worse and if T_d is small then the result becomes reverse.

$$C_{dis} = MIN_{i=1}^{216} \sqrt{\left(P_r - C_{i_R}\right)^2 + \left(P_g - C_{i_G}\right)^2 + \left(P_b - C_{i_B}\right)^2} \quad \ldots \quad \ldots \quad \ldots \quad (2)$$

3.3 Finding the Dominant Colors

We need to find out the dominant colors of an image as well as their percentage in the image. We store this particular information in database against a specific image using the algorithm 3.2. In case of searching an image we compare these information of query image and those of images in database. To find out the dominant colors, at first we have to calculate percentage (*i.e.* number of pixel of that color divided by the total number of pixel in the image) of each color of color

codebook in the image. Then sort these percentage values in descending order and select those colors whose percentage values are greater than T, as dominant colors. Here T is a threshold value and it is determined by the following formula

$$T = \frac{(\text{maximum percentage value} + \text{minimum percentage value})}{2} \quad \ldots \quad \ldots \quad \ldots \quad (3)$$

Here, we give the proposed algorithm [10] for segmentation.

Algorithm 3.2: Fast Image Segmentation Algorithm

1. i. Start from the top left pixel of the image and approach in left to right and top to bottom manner.
   ii. Store the RGB components of the pixels of the image after certain interval T_d.
   iii. We define T_d with respect to the spatial coordinates by our experiment where T_d is a small integer.

2. For each pixel – search the nearest color by comparing the distance (D) between the pixel color (I) and the color in the codebook (C) using the formula no 2.

3.
   i. Assign RGB components of color codebook to the pixel where distance, D minimum.
   ii. Determine the percentage of each color of color codebook in the modified image.
   iii. Select those colors as dominant whose percentage values are more than T using the formula no 3.

3.4 Indexing and Searching Scheme

A given image region is first segmented into a number of dominant regions. Each dominant region is represented by dominant colors. These dominant colors are represented by the index of their closest match in the color codebook C. These index values are then sorted in non-decreasing order and then stored in the database as image features.

A spatial type of indexing [10] is used here for storing and retrieving image features. Color codebook contains 216 colors. We have considered a hash table where there are a total of L levels and k entries. Each i^{th} entry ranges over 54 values, for $1 \leq i \leq k$ and each i^{th} entry has link with another $k-i$ entries and so on. On the basis of the color index values we store these values in specific location of the database. This location is determined by the value of level and bucket. So, similar types of images are stored separately and of course, an image is stored in only one location of the database.

In the case of searching an image we check only the specific location of the database, where similar types of images whose colors have the same level and entry values as those of the query image, are searched. We need not to search all the images stored in the database.

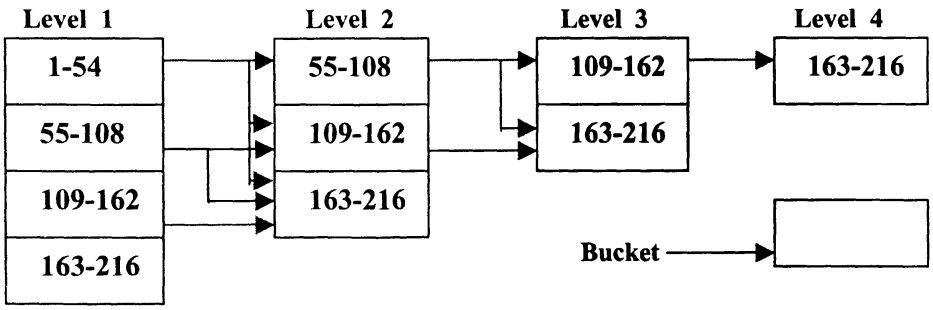

Figure 3.2: Indexing Scheme Using Hash Structure

Here we see from the Figure 3.2 that each bucket contains fifty-four values sequentially. As we work with 216 colors, there is four buckets in the first level. Each first level buckets are linked with the next level that contains the higher values than the previous one. The second level is connected to third level and so on. Here index value means the percentage values of the dominant color and each bucket represents the specific location in the database.

In case of storing the index values at first sort these values in ascending order, then compare the first index value with the values of bucket in the first level and enter into the bucket in which the value is contained. Then compare the second index value whether this value is contained in the same bucket or not. If it is contained then compare the third values whether this value is contained in the same bucket or not. And if the second value is not contained in that bucket then enter into that bucket in the next level where the second index value is contained. Then compare the remaining values similarly and store these entire index values in the specific bucket (*i.e.* in a

specific location in a database) in which the last index value enters. In case of searching an image, similarly we have to find out the specific bucket in which all the index values of the query image are to be stored. Then compare the index values of the query image with those of the images that are stored on the same bucket.

For example suppose the index values of an image are {10, 60, 70, 190}. At first we see in figure 3.3 that 10 is in the range of the first bucket. Then as the second value 60 is not in the range of the same bucket, then we go to the first bucket of the second level.

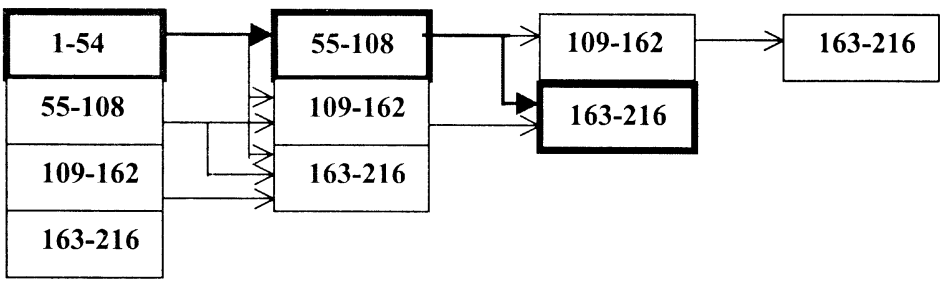

Figure 3.3: Graphical Representation of Algorithm 3.3 for an image whose index values are {10, 60, 70, 190}.

The third value 70 is in the range of the same bucket so we go to the next index value. The next and last value 130 is not in the range of the same bucket, so we go to the next level (*i.e.* 3rd level). We see that 190 is in the range of the second bucket. So we store all the index values in a specific location in database corresponding to the second bucket of the third level.

The algorithm 3.3 takes an array of index values (*i.e.* percentage values of the dominant colors) as input. A special type of hash table is used here as shown in figure 3.1. The hash table contains total of k buckets in the first level and each bucket in the table contains 54 values. Each i^{th} bucket has link with another $k - i$ buckets in the next level for $1 \leq i \leq k$. The output of this algorithm is the number of level and no of bucket in that level to which all the index values have to be stored. Step 1 is the initialization of some variables. Initially the value of the level (variable L) is 1. Step 2 is executed for each index value in the array staring from the first to the last. Step 2.i determines the bucket in which the current index value, X_j is contained. Step 2.2 is only executes for the first index value of the array. Step 2.3 is executed only when the current index value does not contained in the bucket in which the previous index value is contained. This step determines the specific no of bucket in the next level for the current value. Finally step 3 stores

all the index values in a specific location of the database depending the no of level and no of bucket in which the last index value is contained.

Algorithm 3.3: Algorithm for Storing The Color Features in Database

Input: An array $X = \{x_1, x_2, \ldots, x_n\}$ where $x_1 < x_2 < \ldots < x_n$ and n is the total number of elements. A hash table H where

i. there is total of k buckets in the first level

ii. each i^{th} bucket ranges for 54 values in each level, for $1 \le i \le k$

iii. each i^{th} bucket has link with another $k-i$ buckets in the next level and so on.

Output: The number of level and no of bucket in that level to which all the index values have to be stored.

begin

1. **Set** $Z := 1$; $L := 1$;
2. **for** $1 \le j \le n$

begin

i. **Set** $i := \lfloor (X_j - 1) \div 54 \rfloor + 1$; // Determines the bucket in which the current index value, X_j is ranged for.

ii. **if** $Z := 1$ **then** //

Set $m := i$;

Set $Z = 0$;

end if

iii. **if** $m \neq i$ **then**

Set $i := i - L$;

Set $L := L + 1$;

go to the L^{th} level of i^{th} entry.

end if

end for

3. Store all X_j in a specific location in the database. This location is determined by the entries and levels that contain those index values.

end

3.5 Matching Between The Query Image and The Test Image in Database

To find a query image in database we have to calculate the difference between the query image and the test image in the database. If the difference is less than a threshold value then the image is found otherwise not. The perfect value of threshold depends on experiment. If threshold value is large then the number of matched image will be more but the perfection of the result reduces. If threshold is small the number of matched image will be less but the perfection of the result increases.

Let V and V' represents the percentage values of dominant colors of the query image and the test image in the database respectively. Then we compute the difference of the corresponding dominant colors between the query image and the test image and finally sum those differences by the equation 4.

$$Diff = \sum_i (|V_i - V_i'|) \qquad \ldots \ldots \ldots (4)$$

If $Diff < T$ (threshold) then the query image matches with test image otherwise it does not match.

4. Experimental Set Up and Results

We have chosen an image database of 160 fruits, flowers, vegetables and simulated objects (squares, rectangles, triangles, circles, etc). Image segmentation and region feature extraction are done on these images. The proposed method utilizes color information for region search and retrieval. The system is developed in Microsoft Visual Basic 6.0 and Microsoft Access. The number of preferable regions is set to 3-12 for each image. The segmentation is done using only color feature and each segmented region contains only one color. The region information is then stored in the database using hash-indexing structure. This proposed indexing structure [10] shows

better performance in comparison with other systems [8, 11]. In the existing method [8], at first initial segmentation of the image is obtained. It often has over segmented regions. These regions are merged based on their color similarity and continuing in this manner until a specified number of regions are got. So, there is no straightforward method to get dominant regions of images. The combined index used in [9] is a very large number and only applicable when the number of regions are small, only three here. For large number of regions using more color and more shapes the index contains a very large number, which is difficult to maintain and needs more storage and complex type of calculation. All the pixels are considered to make calculations and define the corresponding color class labels [8]. Besides, the input for query is an image, so the computational complexity is greater. So, the time and computational complexity is very high in case of large size images. Whereas in our proposed method we store only color information (*i.e. percentage value and index value of color codebook*) and for query take the same information as input.

The proposed approach requires a low computational cost and results in fast searching and retrieval. Because similar images are stored in the same location of the database and it reduces the searching time as well as improves the retrieval performance greatly. To search for an image it needs not to search for the entire database. We don't calculate all the pixels in the image for segmentation.

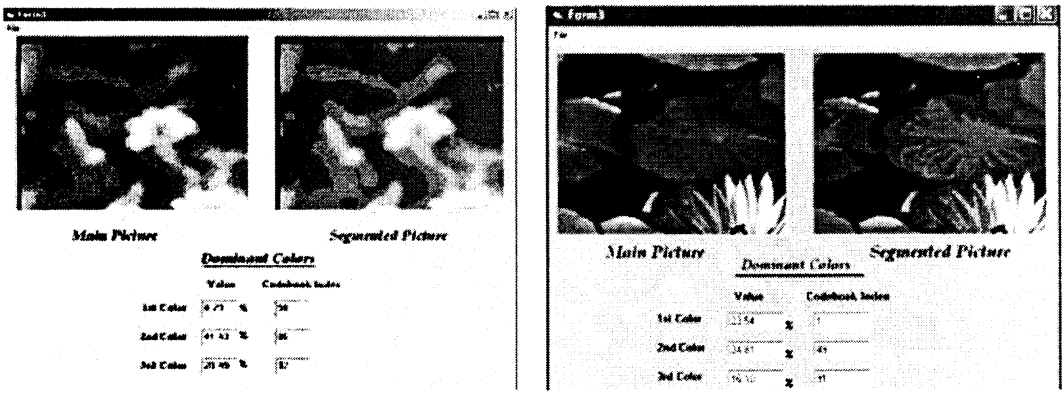

Figure 3.4(a), 3.4(b) Segmentation and Dominant Color Extraction

Figure 3.4 shows the segmentation of a picture by the proposed method. The figure also shows the codebook index and percentage value of the dominant colors of this picture. These values are stored in the database against this picture using the proposed indexing technique.

Here we give some segmented image performed by our proposed method.

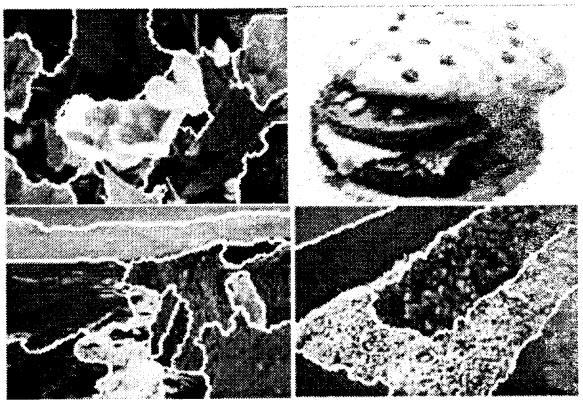

Figure 3.5: Some Segmented Image by the Proposed Method

Figure 3.6 shows searching of the same picture, which used in figure 3.4(a) by using the codebook index and the percentage of the dominant colors of that picture. All the colors of the color codebook are also shown here and the query picture that is found in the database is shown below the query information.

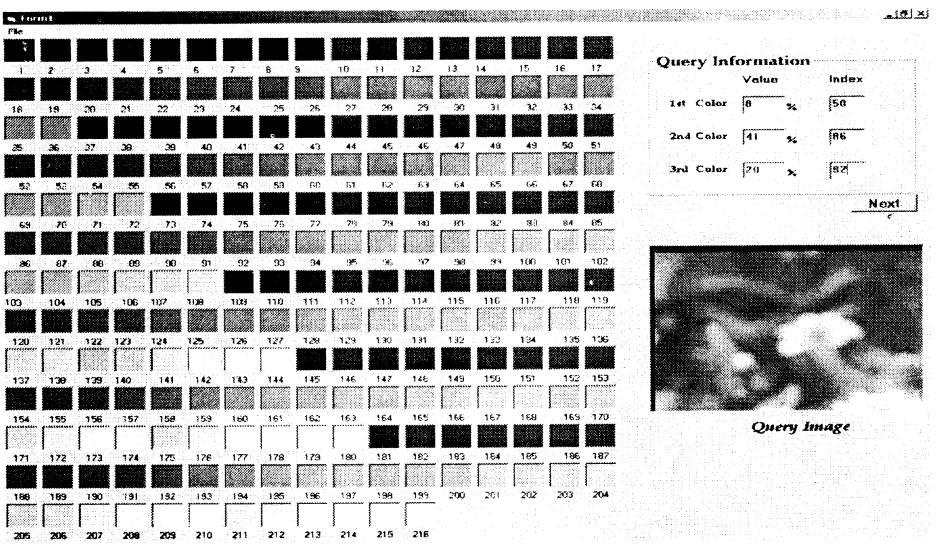

Figure 3.6: Searching a Picture Using the Query Information

Now we see the performance of our proposed technique and the existing method [8]. Consider an image has $n \times n$ pixels and $f(x)$ represents the operation performed by the existing system and the

time required is t_1 and $g(x)$ represents the operation performed by the proposed method and the time required is t_2 then

$$t_1 = \sum_{x=1}^{n \times n} f(x) \text{ and } \quad t_2 = \sum_{x=1}^{p \times p} g(x) \text{ Where } p = n/T_d.$$

Therefore, we see that $t_1 > t_2$. Hence the proposed system requires less time than existing system and becomes faster. The table shows the time required to segment image in proposed system and in existing system.

Table 4.2: Comparison Table of the Existing Segmentation Method and Proposed Method

Number of pixel	Segmentation Time (sec) (Proposed method)	Segmentation Time (sec) (Existing method)
300×300	15	27
250×250	11	21
200×200	7	16
150×150	5	12

Figure 3.7 shows the comparison graph between the existing system [8] and the proposed system where the solid line indicates the proposed system and the dot line indicates the existing system. It is the comparison of the segmentation algorithm. The data is get from Table 4.2 It also shows that the proposed system requires less time to segment an image. The fractional points are omitted here because it is small with respect to the time value given in second.

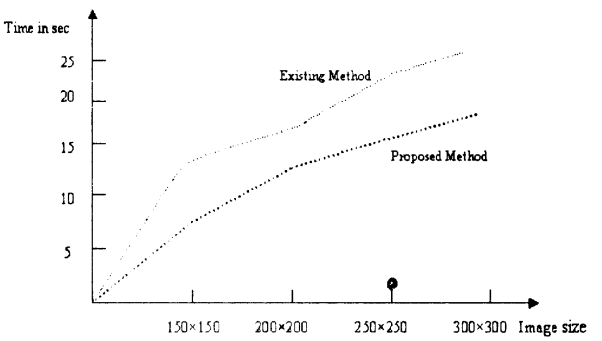

Figure 3.7: Comparison Graph of Existing System and Proposed System

The retrieval accuracy is measured by precision and recall where *Precision (K)* = C_K / K and *Recall (K)* = C_K / M.

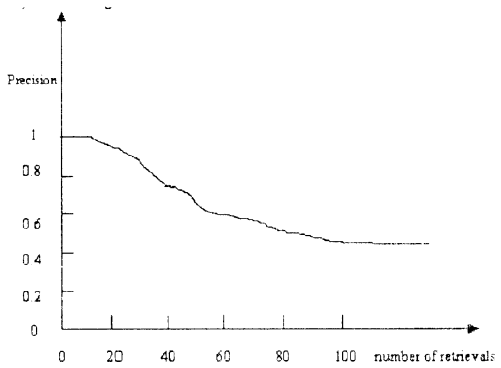

Figure 3.8: Average Precision Versus Number of Retrieval

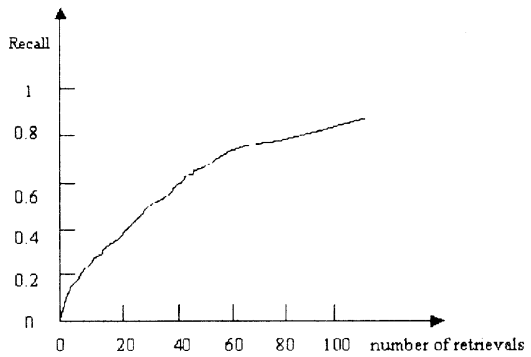

Figure 3.9: Average recall versus number of retrievals

Here K is the number of retrievals, C_K is the number of relevant matches among all the K retrievals, and M is the total number of relevant matches in the database obtained through the subjective testing.

5. Conclusion

We developed a method for storing and retrieving images. To construct this method at first a color codebook of 216 colors is constructed which describes our region color information significantly. This larger number of colors is used so that the images perceptual quality is not affected significantly. Then a compact color feature representation is used which is appropriate for segmented regions. Finally, we described a better approach for image segmentation and searching image regions based on local image properties such as color. Experimental Results shows that our proposed method is faster than the existing method. After segmentation the image

is partitioned into some dominant regions (3-12), which makes the storage task easier and simpler. Here a special type of indexing scheme is used for storing and retrieving images. In the proposed indexing scheme similar images are stored in the same location of the database, which reduces the searching time and improves the retrieval performance.

Although the developed system is successful in many respects, it has some obvious limitations. In terms of uniform distribution of image pixels the pixel colors are distributed coarsely in the entire image. If segmentation is done on this type of images then the number of dominant regions increases and the success rate of this proposed algorithm is reduced in this case.

The system has been implemented with the use of color feature for segmenting and indexing the images. We have developed a faster segmentation method for segmenting the images and a special type of indexing technique for image storing in database. For future enhancement shape and location of the region may be used for getting better performance.

References

[1] M.J. Swain, D.H. Ballard, "Color Indexing", *International Journal of Computer Vision*, Vol. 7, No. 1,P. 11-32,1999.

[2] C. Carson, S. Belongis, H. Greenspan, J. Malik, "*Region-Based Image Querying*", In IEEE Workshop on Content-based Access of Image and Video Libraries, June 1997, pp. 42-49.

[3] Stricker, M. A., and Orengo, M., "*Similarity of color images*," in Proc. SPIE Storage Retrieval Still Image Video Databases IV, vol. 2420, 1996, pp. 381-392.

[4] Stricker, M., and Dimai, A., "*Color indexing with weak spatial constraints*," in Proc. SPIE Storage Retrieval Still Image Video Databases IV, vol. 2670, 1996, pp. 29-40.

[5] Smith, J., and Chang, S. F., "*Tools and techniques for color image retrieval*," Proc. SPIE, vol. 2670, 1996, pp. 2-7.

[6] Pass, G., and Zabih, R., "*Histogram refinement for content based image retrieval*," in Proc. IEEE Workshop Applications Computer Vision, 1996, pp. 96-102.

[7] Huang, J., Kumar, S. R., Mitra, M., Zhu, W., and Zabih, R., "*Image indexing using color correlograms*," in Proc. IEEE conf. Computer Vision Pattern Recognition, 1997, pp. 762-768.

[8] Y. Deng, B.S. Manjunath, H. Shing "*Color Image Segmentation*", Proc of CVPR 1999.

[9] Prasad, B.G., Biswas, K.K., and Gupta, S.K., "*Image Representation using Integrated Color-Shape-Location Index*", NCDAR, Mandya, India, 2001.

[10] Koushik Kanti Mandal, Mr. Rafiqul Islam, Md. Parvez Reza and Fahmida Khatun "*An Efficient Technique for Image Retrieval using Color*" non-published Undergraduate Thesis, Computer Science and Engineering Discipline, Khulna University, Bangladesh.

[11] Hel-Or, H. Z., Yitzhaki, Y., and Hel-Or Y., "*Geometric Hashing Techniques for Watermarking*," Proc. ICIP'01, Greece, pp.498-501, Jan. 2001.