Query Processing for Content Based Image Retrieval

J. Sreedhar, S. Viswanadha Raju, A. Vinaya Babu

***Abstract*—In this paper, we investigated image retrieval based on image content, Content Based Image Retrieval (CBIR) and proposed a framework to characterize the image content and similarity between the images. Our paper discusses the CBIR problem and the solution. Due to the enormous increase in image database sizes, the need for the development of CBIR systems arose. Firstly, this paper outlines feature extraction methods for color and texture. The extracted features for each image in the database used as the basis for similarity between the images. We built user interface based on Java, in which user can easily select query image and view top ten retrieved images based on decreasing order. We extended our approach to sub image retrieval also. Our results report that HSV based color features and contrast based texture features outperform than RGB based features and results reported in the paper are convincing.**

***Index Terms*—Colour, Texture, Query, CBIR.**

I. INTRODUCTION

There has been a rapid increase in the size of digital image collections in the recent years. Everyday, both military and civilian equipment generates giga-bytes of images. A huge amount of information is out there. However, it cannot be accessed or made use of the information unless it is organized so as to allow efficient browsing, searching, and retrieval. There has been a very active research in the area of image retrieval since the 1970s, with the thrust from two major research communities, database management and computer vision. These two research communities study image retrieval from different angles, one being text-based and the other visual-based.

The text-based image retrieval can be traced back to the late 1970s. A very popular framework of image retrieval then was to first annotate the images by text and then use text-based database management systems (DBMS) to perform image retrieval. Representatives of this approach are [1, 2,3,4]. Two comprehensive surveys on this topic are [5,6]. Many advances, such as data modeling, multidimensional indexing, and query evaluation, have been made along this research direction. However, there exist two major difficulties, especially when the size of image collections is large (tens or hundreds of thousands). One is the vast amount of labor required in manual image annotation. The other difficulty, which is more essential, results from the rich content in the images and the subjectivity of human perception. That is, for the same image content different people may perceive it differently. The perception subjectivity and annotation impreciseness may cause unrecoverable mismatches in later retrieval processes. In the early 1990s, because of the emergence of large-scale image collections, the two difficulties faced by the manual annotation approach became more and more acute. Content-based image retrieval was propose, to overcome these difficulties. That is, instead of being manually annotated by text-based key words, images would be indexed by their own visual content, such as colour and texture. Since then, many techniques in this research direction have been developed and many image retrieval systems, both research and commercial, have been built. The advances in this research direction are mainly contributed by the computer vision community. Many special issues of leading journals have been dedicated to this topic [7,8,9,10,11]. Content-based image retrieval, we feel there is a need to survey what has been achieved in the past few years and what are the potential research directions which can lead to compelling applications. Since excellent surveys for text-based image retrieval paradigms already exist [5, 6], in this paper we will devote our effort primarily to the content-based image retrieval paradigm. There are three fundamental bases for content-based image retrieval, i.e. visual feature extraction, multidimensional indexing, and retrieval system design. The fundamental difference between content-based and text-based retrieval systems is that the human interaction is an indispensable part of the latter system. Humans tend to use high-level features (concepts), such as keywords, text descriptors, to interpret images and measure their similarity. While the features automatically extracted using computer vision techniques are mostly low-level features (colour, texture, shape, spatial layout, etc.). In the past decade, a few commercial products and experimental prototype systems have been developed, such as QBIC [4], Photobook [5], Virage [6], VisualSEEK[7], Netra [8], SIMPLIcity [9].

II. RELATED WORK

Due to the high ambiguity involved in interpreting visual information on the affective level, it is very difficult to predict human emotions from visual information. As such, determining the relation between human emotions and visual information is very important. Generally, images provide colour, texture, shape, and pattern information.

Manuscript Received October 18, 2011.

J. Sreedhar, Associate Professor in Computer Science & Engineering,, DVRCET, Hyderabad, India, Mobile No: 8790423564, (e-mail: sreedharyd@gamil.com).

Dr. S. Viswanadha Raju, Professor in CSE, SIT, JNT University, Hyderabad, India, 9963701506, (e-mail: svraju.jntu@gmail.com).

Dr. A. Vinaya Babu, Professor in CSE & Director of Admissions, JNT University, Hyderabad, India.

The colour feature is one of the most widely used visual features in image retrieval. It is relatively robust to background complication and independent of image size and orientation. Some representative studies of colour perception and colour spaces can be found in [12,13, 14]. In image retrieval, the colour histogram is the most commonly used colour feature representation. Statistically, it denotes the joint probability of the intensities of the three colour channels. Swain and Ballard proposed histogram intersection, an L1 metric, as the similarity measure for the colour histogram [15]. To take into account the similarities between similar but not identical colours, Ioka [16] and Niblack et al. [17] introduced an L2-related metric in comparing the histograms. Furthermore, considering that most colour histograms are very sparse and thus sensitive to noise, Stricker and Orengo proposed using the cumulated colour histogram. Their research results demonstrated the advantages of the proposed approach over the conventional colour histogram approach [18]. Besides the colour histogram, several other colour feature representations have been applied in image retrieval, including colour moments and colour sets. To overcome the quantization effects, as in the colour histogram, Stricker and Orengo proposed using the colour moments approach. The mathematical foundation of this approach is that any colour distribution can be characterized by its moments. Furthermore, since most of the information is concentrated on the low-order moments, only the first moment (mean), and the second and third central moments (variance and skewness) were extracted as the colour feature representation.

To facilitate fast search over large-scale image collections, Smith and Chang proposed colour sets as an approximation to the colour histogram [19, 20]. They first transformed the (R, G, B) colour space into a perceptually uniform space, such as HSV, and then quantized the transformed colour space into M bins. A colour set is defined as a selection of colours from the quantized colour space. Because colour set feature vectors were binary, a binary search tree was constructed to allow a fast search. The relationship between the proposed colour sets and the conventional colour histogram was further discussed [19, 20].

III. OVERVIEW OF PROPOSED APPROACH

A. Solution

The solution proposed is to extract the primitive features of the query image and compare them to those of database images. The image features under consideration are colour (in the HSB colour space) and texture. For the purpose of querying with respect to a part (or parts) of the query-image, each image in the database is divided into nine sections as shown infigure1. Each section is represented by a vector which consists of a total of 18 values, of which the first 6 values represent the average hue, average saturation, average brightness, variance of hue, variance of saturation and variance of brightness respectively. The next 12 values of each section vector represent the texture attributes. Thus, each image in the database is stored as a vector of 162 floating point values (or 9 sections, each section being represented by a vector of 18 floating point values).

Table1. Image subsections Matrix

1	2	3
4	5	6
7	8	9

When a query-image is presented, if the query is with respect to the whole image, the distance of each image in the database with respect to the query-image is calculated, and the top five images with the least distance with the query-image are displayed as results. If the query is with respect to a section (or sections) of the query-image, then these distance metrics are calculated with respect to sections only. The user can select the way he wishes to query a sub-image. Finally, resultant images are displayed based on increasing similarity.

IV. IMAGE CONTENT CHARACTERIZATION

A. Colour representation

One of the most important features that make possible the recognition of images by humans is colour. Colour is a property that depends on the reflection of light to the eye and the processing of that information in the brain. We use colour everyday to tell the difference between objects, places, and the time of day. Usually colours are defined in three dimensional colour spaces. These could either be RGB (Red, Green, and Blue), HSV (Hue, Saturation, and Value) or HSB (Hue, Saturation, and Brightness). Thus, a vector with three co-ordinates represents the colour in this space.

B. Motivation for using HSV Color Space

Hue, Saturation and Value are based on the artist concepts of Tint, Shade, and Tone, respectively. The HSV colour space is really a cylinder, and not a cone or hexcone as usually pictured. However, the perceived change in colour as saturation varies between 0 and 1 is less for dark colours (i.e. ones with a low Value parameter) than for light ones (i.e. ones with a high Value parameter), so the colour space is usually distorted to form a cone to help compensate for this perception imbalance (although the space is still not perceptually uniform -- for an example of a perceptually uniform space, see the CIE LUV colour space).The HSV colour space, like RGB, is a device-dependent colour space, meaning the actual colour you see on your monitor depends on what kind of monitor you are using, and what its settings are Complementary colours in the hexcone are 180 degrees opposite one another. One of the first colour systems based on polar coordinates and perceptual parameters was Munsell's (A Colour Notation. Baltimore MD: Munsell Colour Co. 1946), which defined the terms: "Hue: It is that quality by which we distinguish one colour family from another, as red from yellow, or green from blue or purple." "Chroma [Saturation in HSV]: It is that quality of colour by which we distinguish a strong colour from a weak one; the degree of departure of a colour sensation from that of a white or gray; the intensity of a distinctive hue; colour intensity." "Value: It is that quality by which we distinguish a light colour from a dark one."

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

The advantage of HSV colour space is its ability to separate chromatic and achromatic components. Therefore we selected the HSV colour space to extract the colour features according to hue, saturation and value. HSV colour space is widely used in computer graphics, visualization in scientific computing and other fields [21,22].

The proposed method used HSV(Hue, Saturation and Value) Colour space, because it is natural and is approximately perceptually uniform.

Traditionally, the main method of representing colour information of images in CBIR systems has been through colour histograms. But in our approach, we have represented colour in the HSB colour space, as floating point values. Each section in an image has six values representing colour. These six values represent the average hue, average saturation, average brightness, hue variance, saturation variance and brightness variance respectively, for that section.

$$H = \begin{cases} 60\left(0 + \dfrac{G-B}{C_{\max} - C_{\min}}\right), \text{if } C_{\max} = R \\ 60\left(2 + \dfrac{B-R}{C_{\max} - C_{\min}}\right), \text{if } C_{\max} = G \\ 60\left(4 + \dfrac{R-G}{C_{\max} - C_{\min}}\right), \text{if } C_{\max} = B \end{cases} \quad (1)$$

$$S = \frac{C_{\max} - C_{\min}}{C_{\max}} \quad (2)$$

$$V = C_{\max} \quad (3)$$

Where $C_{\max}$ = MAX (R, G, B) and

$C_{\min}$ = MIN (R, G, B)

C. Texture Representation

Because the final formatting of your Texture is that innate property of all surfaces that describes visual patterns, each having properties of homogeneity. It contains important information about the structural arrangement of the surface, such as; clouds, leaves, bricks, fabric, etc. It also describes the relationship of the surface to the surrounding environment.

The ability to retrieve images on the basis of texture similarity may not seem very useful. But the ability to match on texture similarity can often be useful in distinguishing between areas of images with similar colour (such as sky and sea, or leaves and grass). A variety of techniques has been used for measuring texture similarity; the best-established rely on comparing values of what are known as second-order statistics calculated from query and stored images. Essentially, these calculate the relative brightness of selected pairs of pixels from each image. From these it is possible to calculate measures of image texture such as the degree of contrast, coarseness, directionality and regularity, or periodicity, directionality and randomness.

In our approach, while the first six values in each section vector represent colour, the next twelve values represent texture. Now, these values are calculated iteratively by finding the characteristics for each overlapping 5 x 5 contrast element matrix, along 0 degree, 45 degrees, 90 degrees and 135 degrees, within each section. The meaning of these characteristics along different angles are: Along x-axis => 0 degree, Along x=y axis => 45 degree, Along y-axis => 90 degree, Along x=-y axis => 135 degree.

Along each angle, values are calculated so as to indicate the number of 5 x 5 pixel regions in that section, where the upper half of that region is brighter (or darker) than the lower half, where the left half of that region is brighter (or darker) than the right half, where the half to the left diagonal is brighter (or darker) than the other half, where the half to the right diagonal is brighter (or darker) than the other half, and where the two portions in consideration are almost same(with a 10% difference).

When all these calculations are complete, we get our twelve texture features representing the total number of bright, equal and dark regions along the four different angles.

Thus, now the feature vector for one section of an image is computed, and it is represented by 18 values. Similarly, the vectors are computed for all the remaining eight sections. Thus, the feature vector for an image comprises (18 x 9) 162 values. Similarly, the feature vectors for all other images in the database are computed and stored.

He and Wang [23] determine a texture spectrum using 3×3 windows. The gray level of the central pixel is compared with the other eight pixels in the window. Each pixel is assigned a value of 0 if its value is less than, 1 if its value is equal to, and 2 if its value is greater than that of the central pixel. The central pixel is not assigned any value. Using such a scheme, the number of gray levels is reduced to 3. After the reduction, the number of all combinations within the 3 × 3 window is 38 = 6561. He and Wang proposed a simple scheme to assign a number between 0 and 6560 automatically to each possible pattern of 0s, 1s and 2s in a window. We now extend the same idea to a larger, and visually more meaningful 5×5 window. The number of possible patterns is a prohibitive 324 if we blindly follow He and Wang's approach. Our idea is to reduce the 5 × 5 window to a 3 × 3 window. Let W be a 5 × 5 block of pixels. We use only 17 out of the 25 pixels, indexed as shown in Fig. 1, in our reduction. These pixels are in the eight compass directions with respect to the central pixel ac. The average intensity of the pixels along each direction is compared with that of the central pixel in the reduction. The reduction of a 5×5 window W into a 3×3 block U is given.

$$\begin{pmatrix} a_0 & - & a_2 & - & a_4 \\ - & a_1 & a_3 & a_5 & - \\ a_{14} & a_{15} & a_c & a_7 & a_6 \\ - & a_{13} & a_{11} & a_9 & - \\ a_{12} & - & a_{10} & - & a_8 \end{pmatrix}$$

Fig. 1. Pixel index values

mathematically by

$$a_i^U = \begin{cases} 2, \text{if } a_{2i}^W + a_{2i+1}^W - 2a_c^W > 0 \\ 1, \text{if } a_{2i}^W + a_{2i+1}^W - 2a_c^W = 0, \ 0 \le i \le 7 \\ 0, \text{if } a_{2i}^W + a_{2i+1}^W - 2a_c^W < 0 \end{cases} \quad (4)$$

with the superscripts on the right hand sides indicating that the pixels are taken from the 5 × 5 window W. The reduced block U is identical to the 3×3 window used by He and Wang and we apply the same method proposed by them to generate a unique texture unit number

$$\tau(U) = \sum_{i=0}^{7} a_i^U 3^i \quad (5)$$

For any M×N pixel image, there will be (M−2)×(N−2) intensity patterns of size 5 × 5, each described uniquely by τ (U) such that $0 \le \tau(U) \le 6560$. Texture Unit Spectrum, TUS (t, $0 \le t \le 6560$), is the histogram of τ (U)s within an image. Several useful features, corresponding to visually meaningful patterns, may be defined on the TUS.

i. Surrounding contrast

Surrounding contrast (sc) features define the set of patterns that have a uniform neighbourhood around the central pixel. All the pixels in the neighbourhood have the same property with respect to the central pixel, they are all brighter, darker or equally bright as the central pixel. These three neighborhoods are shown in Fig. 3: (a) defines the pattern where the surrounding pixels are all brighter, (b) the pattern where the surrounding pixels are equally bright, and (c) the pattern where the surrounding pixels are darker than the central pixel. Sc_bright feature measures the frequency of occurrence of pattern in Fig. 3(a) in the image and is given by TUS(t = 6560). Sc_equal and sc dark are similarly defined based on the patterns in Fig. 3(b) and (c) respectively and are given by TUS(3280) and TUS(0).

ii. Alternating contrast

Alternating contrast (ac) features measure the frequency of occurrence of local patterns in which the brightness of the surrounding pixels alternates between being brighter and being darker than the central pixel. There are six such possible configurations as shown in Fig. 4.

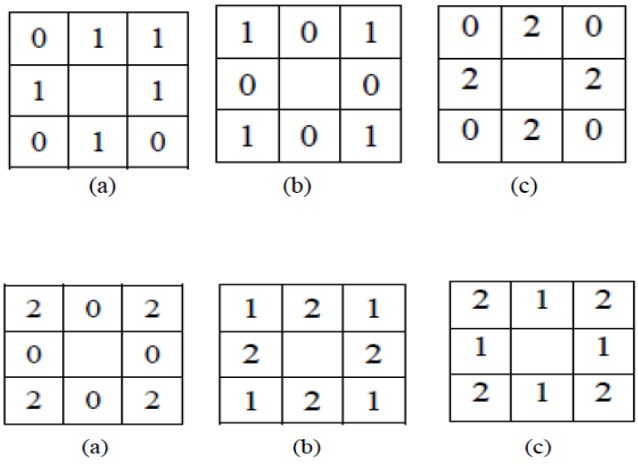


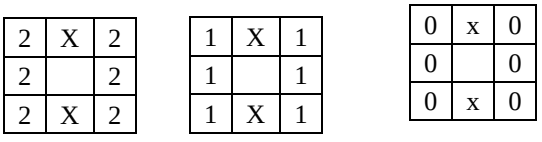
Fig. 3. Alternating Contrast Patterns

These are grouped into two groups of three each. ac_bright is the collection of patterns in which the 4-connected neighbours are brighter than the remaining pixels while ac_dark refers to alternating patterns in which they are darker. ac_bright consists of patterns (a), (c) and (e) given by TUS(2460), TUS(4920) and TUS(5740) respectively. ac dark refers to patterns (b), (d) and (f) given by TUS(820), TUS(1640) and TUS(4100) respectively.

ac_bright = TUS(2460) + TUS(4920) + TUS(5740)
ac_dark = TUS(820) + TUS(1640) + TUS(4100)

iii. Vertical and horizontal contrasts

Vertical contrast (vc) feature measures three groups of vertical stripes, vc_bright, vc_equal and vc_dark shown in Figs. 5(a), (b) and (c) respectively. x is one of 0, 1 or 2. In each group, one of the possible values of x makes the configuration identical to surrounding contrast feature. For example, if x = 2 in Fig. 5(a), then the configuration is the same as that of sc_bright. Therefore, the possible number of patterns for vc_bright, vc_equal and vc_dark each is two and



(a) (b) (c)

Fig. 4. Vertical Contrast Patterns

vc feature is a summation of the frequencies of occurrence of these six patterns.

vc_bright = TUS(6068) + TUS(6314)
vc_ equa l = TUS(3034) + TUS(3526)
vc _dark = TUS(246) + TUS(492)

Horizontal contrast (hc) measures the frequency of occurrence of horizontal stripes in an image. The brightness of the central pixel is compared against that of the pixels in the previous and succeeding rows. If the neighbouring pixels are all brighter than the central pixel, then we define a *hc_bright* pattern. Similarly, we define *hc_equal* and *hc_dark* patterns. The three patterns and possible configurations are 90◦ rotated versions of vertical contrast patterns. The horizontal contrast features are computed from the TUS as

hc_bright = TUS(2132) + TUS(6533)
hc _equal = TUS(1066) + TUS(5494)
hc_dark = TUS(2214) + TUS(4428)

In summary, we define four sets of useful patterns easily computable from the texture unit spectrum, viz., surrounding contrast, alternating contrast, vertical and horizontal contrasts. In each category, with the exception of alternating contrast, there are three sub-classes of patterns depending on whether the neighbouring pixels are brighter, equal in brightness, or darker than the central pixel. There are only two sub-classes in alternating contrast feature and that gives a total of 11 features based on the visually observable texture patterns in 5 × 5 windows.

iv. Contrast Patterns In 5 × 5 Windows

In this section, we define a new set of patterns on 5 × 5 windows based on contrast variations rather than intensity. We consider the four major directions: 0^0, 45^0, 90^0 and 135^0. Contrast is measured by the difference in the sums of intensities of pixels lying on either side of a line drawn in the specified direction through the centre of the 5×5 window. For example, contrast at 45^0 is measured as shown in Fig. 6. Ai are always to the left as we walk in the direction of the line. Contrast is then reduced to three categories: bright, equal or dark depending on whether it is greater than, equal to or less than 0. In practice, contrast is considered equal if the difference is less than a predefined tolerance factor expressed as a percentage of a_i. Thus, each window in the image is characterized by the four directional contrast categories. Contrast features for the image are given by the frequency of occurrence of each of the 12 categories: three each (bright,equal and dark) in the four directions, and thus form a 12- dimensional feature vector. In an M × N pixel image, there are (M−2)×(N−2) windows each contributing to all the four contrast features such that the sum of frequencies of bright,equal and dark categories for any direction is (M−2)×(N −2).

Table 4 : Matrix Values

a_1	a_2	a_3	a_4	-
a_5	a_6	a_7	-	b_1
a_8	a_9	-	b_2	b_3
a_{10}	-	b_4	b_5	b_6
-	b_7	b_8	b_9	b_{10}

$$c_{45} = \sum_{i=1}^{10} a_i - \sum_{i=1}^{10} b_i \qquad (6)$$

Fig. 6. Calculating contrast at 45^0

Along each angle, values are calculated so as to indicate the number of 5 x 5 pixel regions in that section, where the upper half of that region is brighter (or darker) than the lower half, where the left half of that region is brighter (or darker) than the right half, where the half to the left diagonal is brighter (or darker) than the other half, where the half to the right diagonal is brighter (or darker) than the other half, and where the two portions in consideration are almost same(with a 10% difference). When all these calculations are complete, we get our twelve texture features representing the total number of bright, equal and dark regions along the four different angles. Thus, now the feature vector for one section of an image is computed, and it is represented by 18 values. Similarly, the vectors are computed for all the remaining eight sections. Thus, the feature vector for an image comprises (18 x 9) 162 values. Similarly, the feature vectors for all other images in the database are computed and stored.

D. Similarity Measures

The distance between feature vectors is calculated using two methods:

- Square root (Euclidean distance)
- Absolute value (Manhattan distance)

The results produced by these two methods can be different, hence the results produced by each method are displayed to the user is separate areas.

If q = 1, d is Manhattan distance

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ip} - x_{jp}| \qquad (7)$$

If q = 2, d is Euclidean distance

$$d(i,j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots\ldots + |x_{ip} - x_{jp}|^2)} \qquad (8)$$

Where i = (xi1, xi2, …, xip) and j = (xj1, xj2, …, xjp) are two p-dimensional data objects, and q is a positive integer.

V. RESULTS

A. Query-By-Example : Whole Image:

When the user opts for Query-By-Example, he chooses or presents an example-image, or a query-image. The query can now relate to the whole image, or only to a section or sections of the image. In all cases, the feature-vector representing the query image is automatically retrieved from the database values. If the query is with respect to the whole image, all the 162 feature values of each image in the database are retrieved and used to calculate the distance of the database image with the query image. Finally, when the distances for all the images in the database are calculated, the top 7 images with the least distance are displayed as results.

As such, for the whole-image query, very few inconsistencies have been reported. The approach works very well for finding similar images, when considering the whole images except for a very rare occurrence of a not-so-relevant image.

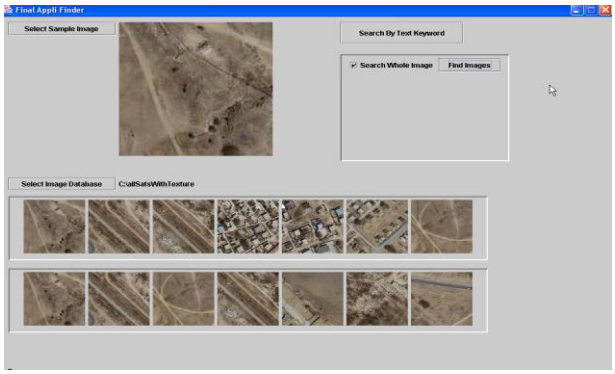


Fig 5:Results of Query-By-Image, when query is with respect to the whole image

When the user clicks on the "Find Images" button, the similarity metrics are performed, and the results are displayed to the user in two separate panes. The results in the first pane are the matches according to square root method, and those in the second pane are according to absolute value method.

B. Query-By-Example : Part or parts of image

If the user wishes to query with respect to sub-images, i.e., part (or parts) of image, the following five options are presented.

- One selected section of query-image Vs All sections of database-images
- One selected section of query-image Vs All sections of database-images, as well as the database-image as a whole
- An adjacency-pattern of query-image Vs All possible adjacency patterns in the database-images
- A two-section-diagonal-pattern in query-image Vs All possible two-section-diagonal-patterns in the database-images

In the case of sub image retrieval, query-image is visually divided into nine sections, or sub-images. The user can select the sections with regard to which the images are to be compared. In case of the first two options, the user should select only one section in the query image. For the third option, user should select exactly two sections, which may or may not be adjacent to each other. For the fourth option, diagonal sections (which should essentially be adjacent) should be selected. Violation of any of these Query-image-section-selection rules produces results which are not correct. For the ease of query-processing with respect to sections, each section is represented as a Rectangle object.

C. Query Processing for One Section Vs All Sections

If the user selects one section in the query-image, and wishes to find those images in the database in which the features of any one of the nine sections are similar to those of the selected section of the query-image. The features of the selected section are retrieved by just identifying the index of the section which is selected.

For each image in the database, distance of each section with the selected section of the query-image is calculated, and the distance value of the section with the least distance is saved.

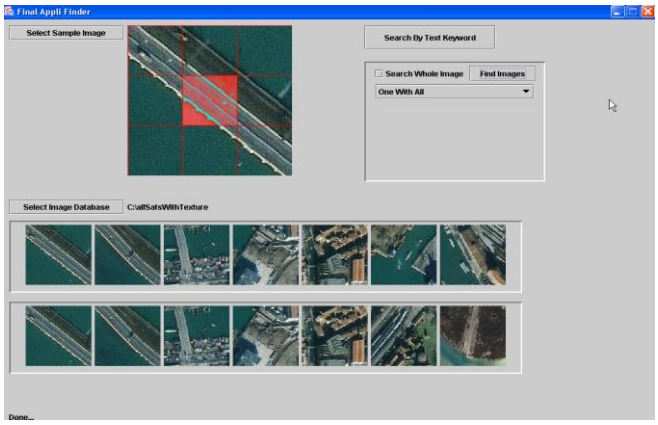


Figure 6 : Results of Sub-image query, w. r. t. one selected section of the query image

One section of the query image is to be selected by the user, and then the user has to click on the "Find Images" button. If the user selects more than one section for this query, the results displayed will be misleading.

D. Query Processing for One Section Vs All Sections and Whole-Image

In this section, we compared with all the nine sections of each image in the database, and also the with the image as a whole, as there may be cases in which the selected section of the query image can be in the form of images in which that selected section itself appears spanning the entire image.

Consider a query-image and a section of that image is selected by the user, to find images in the database, in which either a section is similar to the selected section or as such, the image as a whole is more similar to the selected section, than the individual sections that make up the image. To process such query, consider a database image D1, comprising nine section s1, s2, s3 … s9. The distance of each section, s1 thru s9, with the selected section is computed, and the section si with the least distance is temporarily chosen as the representative distance (tmpDist) for that image D1. Further, the feature vector of 162 floating point values representing D1 is normalized to a feature vector of just 18 features, by summing up all values representing the same feature and dividing the sum by 9. Now, the distance of this normalized feature vector with the selected section feature vector is computed. If this distance is less than the tmpDist computed above, then it means that this image D1, as a whole, is more similar to the selected section of the query-image, than the individual sections of D1. An example of the application of this query can be as follows:

A car appears in a section of a query-image. There might be many images in the database, which have a car. Besides, there can be an image such that the entire image is just a car,

each section representing just a part of the car. To enhance the chance of such database-image to be retrieved as a result, the above functionality is provided. The following test result represents the method called when the user query is with respect to one section of query-image and all sections of the database images, as well as whole images.

It has been surprisingly observed that the results produced by this query are just the same as those produced by the query relating to one section vs. all sections. After a brief discussion, we were able to conclude that in any image in the database, the image whose normalized features are more close to the features of the selected section of the query-image is essentially close to the selected section because of the presence of a section which is very close to the selected section.

The user has to select a query from the list available and the number of sections selected should be in conformance with the type of query selected. Else, the results will be wrong. For example, if the user selects "One with all" query, then he should select only one section in the query-image.

E. Query processing for two selected sections of query-image Vs All possible adjacent two section pairs in the database-images

This query and the next query can process requests where the user is trying to find images in the database in which a specific pattern from two or more selected sections of the query image can be found. The use of this particular query is to find more subtle patterns in the database-images.

In the query-image, the user selects two non-adjacent (or adjacent) sections, and desires to find images in the database in which these two sections appear adjacently. For example, the user selects two sections in query image, such that one section has water, and the other section has some buildings. The interpretation of this query is that the user wishes to find images in the database such that water and buildings appear in adjacent sections. The important characteristic of this particular query is that the adjacency can be between sections. As such, for an image of 9 sections as follows, there can be 20 pairs of adjacent sections which are mentioned in Table1.

From the above diagrammatic representation of the nine sections of an image, it can be easily observed that the sections which are adjacent to each other and form pairs are as follows: For example, section 2 , section 4 and section 5 are adjacent to section 1. Similarly adjacent sections are calculated for all sections.

It is worth noticing that a pair of section1 and section2 is as good as a pair of section2 and section1 and hence only one such pair is considered.

For any two selected sections of the query image, (adjacent or non-adjacent) this query processing involves the comparison of the aggregated features of the two selected sections of the query-image with all the possible 20 pairs of adjacent sections of each database image. When the user selects the two sections, the aggregate features of these selected sections are calculated by just adding up the respective features of the same index. Thus, we have an array of 18 floating point values representing the selected sections.

Consider an image in database as D1. There can be 20 pairs of adjacent sections (as mentioned above). For each pair of adjacent sections in the database image, we compute an aggregate feature vector as representing those two adjacent sections. Thus, we have 20 such feature vectors for D1. Of these twenty pairs of adjacent sections, we find the one with the least distance with respect to the selected adjacent sections of the query image. This operation is performed for each image in the database, and the top 5 database-images with the least distance representative values are retrieved and displayed to the user as matching images to his query.

The following test result represents the output produced when the user query is with respect to two sections of query-image and all possible two-adjacent sections of the database images. Here again, it is imperative that the user select exactly two sections. The misleading results produced in case of selecting more or less than two sections are summarized in the following test results description.

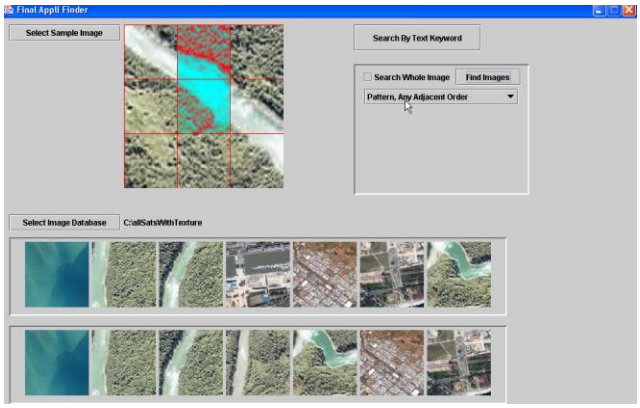


Figure 7 : Results of Sub-image query, w.r.t. two selected adjacent sections of the query-image

This screen appears when the user chooses the "Diagonal Pattern, Occurring Diagonally" option from the ComboBox, selects two diagonally adjacent sections in the query-image, and clicks on the "Find Images" button. If the user selects two sections, which are not diagonally adjacent, it is sure to produce wrong results.

F. Query Processing for a two-section-diagonal-pattern in query-image Vs all possible two-section-diagonal-patterns in the database-images

This query is more specific than the previous query where the pattern of the selected sections of the query-image can appear in any pair of adjacent sections in a database image. This query finds patterns only in diagonals, and the selected pattern of the query image should also be a diagonal pattern. To make it clear, the query image (and in fact any image ion the database) can have a two-section diagonal selection.

In the query-image, the user selects two essentially adjacent sections, and that too sections which are in diagonal order, and desires to find images in the database in which these two sections appear as diagonal sections. For example, the user selects two sections in query image, such that one section has water, and the other section has some buildings. The interpretation of this query is that the user wishes to find images in the database such that water and buildings appear in diagonally adjacent sections. The

important characteristic of this particular query is that the diagonal adjacency can be between sections. As such, for an image of 9 sections as follows, there can be 8 pairs of diagonally adjacent sections which are mentioned in Table1.

From the figure 7, representation of the nine sections of an image, it can be easily observed that the sections which are adjacent to each other and form diagonal pairs are as follows , for example , section 5 is diagonal section for section 1. Similarly diagonal sections are calculated for sub image blocks.

It is worth noticing that a pair of section1 and section5 is as good as a pair of section5 and section1 and hence only one such pair is considered.

For any two selected sections of the query image, (essentially diagonally adjacent) this query processing involves the comparison of the aggregated features of the two selected sections of the query-image with all the possible 8 pairs of diagonally adjacent sections of each database image. When the user selects the two sections, the aggregate features of these selected sections are calculated by just adding up the respective features of the same index. Thus, we have an array of 18 floating point values representing the selected sections.

Consider an image in database as D1. There can be 8 pairs of diagonally adjacent sections (as mentioned above). For each pair of diagonally adjacent sections in the database image, we compute an aggregate feature vector as representing those two adjacent sections. Thus, we have 8 such feature vectors for D1. Of these eight pairs of diagonally adjacent sections, we find the one with the least distance with respect to the selected adjacent sections of the query image. This operation is performed for each image in the database, and the top 5 database-images with the least distance representative values are retrieved and displayed to the user as matching images to his query.

The following test result represents the output produced when the user query is with respect to two diagonally adjacent sections of query-image and all possible two-diagonally adjacent sections of the database images. Here again, it is imperative that the user select exactly two sections, and these selected sections be diagonally adjacent. The misleading results produced in case of selecting more or less than two sections or selecting sections which are not diagonally adjacent are summarized in the following test results description.

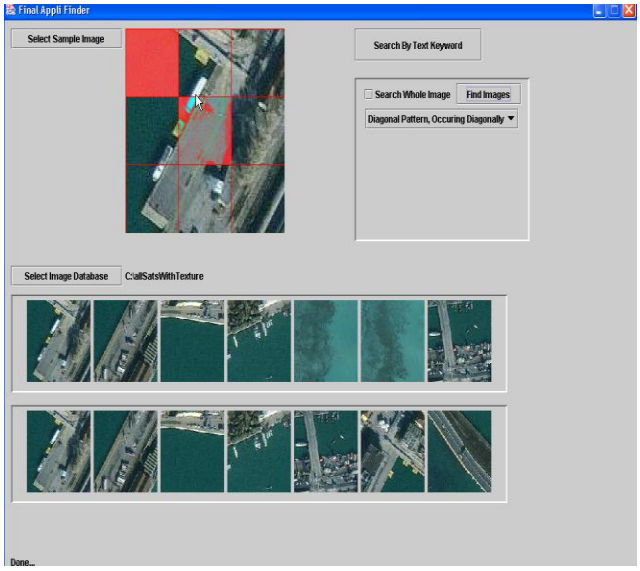


Fig 8: Results of Sub-image query, w.r.t. two selected diagonally-adjacent sections of the query-image.

This particular test result produces null results. This is the only query which actually produces “null” as results in response to wrong selection of sections by the user. Results of Sub-image query, w.r.to. two selected sections of the query-image.

The two sections selected by the user are compared with all possible two-adjacent section pairs in the database. This type of query can enhance our capability to locate patterns.

Table 3 .Performance of different colour spaces for a test set of 10 images taken from the first database.

Query Image	No. Of images Retrieved	RGB		HSV	
		No.of relevent Images retrieved	Precision (%)	No.of relevent Images retrieved	Precision (%)
Query 1	24	16	62.5	19	77.66
Query 2	24	20	68.83	22	80.00
Query 3	24	18	65.00	20	78.00
Query 4	24	20	63.33	18	76.33
Query 5	24	15	60.83	20	78.32
Query 6	24	20	71.63	21	79.33
Query 7	24	14	50.33	18	77.00
Query 8	24	20	65.83	22	80.10
Query 9	24	17	57.50	20	78.86
Query10	24	15	52.00	20	78.83

Table 4: the average precision using RGB, HSV and colour spaces for the two data sets used in testing.

	Average Precision (%)	
	RGB	HSV
Dataset1	67.08	80.15
Dataset2	67.04	78.88

In general, we found that the three colour spaces give similar performance. Precision of a CBIR system is defined as the ratio of the number of relevant images retrieved to the number of images retrieved.For example, in Figure 3, the number of images retrieved is 24 while the number of relevant images is 22.Precision is, thus, 22/24 = 0.92. It is often expressed as a percentage, and then precision is 92%.

Table 2 gives the performance of different colour spaces for a test set of 10images taken from the first database. Table 3 gives the average precision using RGB, HSV and colour spaces for the two data sets used in testing. It is seen that the performance of HSV spaces are comparable to RGB space. As RGB space is primarily developed for modeling colour hardware capabilities and the other two spaces are more closely related to human perception of colour, the result is somewhat unexpected. However, closer examination of the results indicate that HSV space in particular spaces give higher rank to images that are more similar to the query image.

VI. CONCLUSION

Our proposed application performs a simple color and texture-based search in the image database for an input query image. HSV based color features and local contrast varying texture features are extracted for each image in the entire database. Most similar images are retrieved based on distance metrics like Euclidian and Manhattan distance. The retrieved images are displayed in decreasing order of similarity. We compared our retrieval results with RGB and HSV color features along with texture features. Our results report that HSV based color features outperform than RGB features. The average retrieval performance for HSV based features tested on database1 of size 5000 images is reported as 80.15%. Although, we have not evaluated for sub image retrieval performance but results are encouraging as shown in the paper. We conclude that HSV color space mimic human perception of colors so HSV color space features are best suited for applications like CBIR where human evaluation is involved.

REFERENCES

1. N. S. Chang and K. S. Fu, A Relational Database System for Images, Technical Report TREE 79-28, Purdue University, May 1979.
2. N. S. Chang and K. S. Fu, Query-by pictorial-example, IEEE Trans. on Software Engineering SE-6(6), 1980.
3. S.-K. Change Pictorial database systems, IEEE Computer, 1981.
4. S.-K. Chang, C. W. Yan, D. C. Dimitroff, and T. Arndt, An intelligent image database system, IEEE Trans. Software Eng. 14(5), 1988.
5. H. Tamura and N. Yokoya, Image database systems: A survey, Pattern Recognition 17(1), 1984.
6. S.-K. Chang and A. Hsu, Image information systems: Where do we go from here? IEEE Trans. On Knowledge and Data Engineering 4(5), 1992.
7. V. N. Gudivada and J. V. Raghavan, Special issue on content-based image retrieval systems, IEEE Computer Magazine 28(9), 1995.
8. A. Pentland and R. Picard, Special issue on digital libraries, IEEE Trans. Patt. Recog. and Mach. Intell.,1996.
9. A. D. Narasimhalu, Special section on content-based retrieval, Multimedia Systems, 1995.
10. Special issue on visual information management (R. Jain, Guest Ed.), Comm. ACM, Dec. 1997.
11. B. Schatz and H. Chen, Building large scale digital libraries, Computer, 1996.
12. C. S. McCamy, H. Marcus, and J. G. Davidson, A colour-rendition chart, Journal of Applied Photographic Engineering 2(3), 1976.
13. M. Miyahara, Mathematical transform of (r,g,b) colour data to munsell (h,s,v) colour data, SPIE Visual Commun. Image Process. 1001, 1988.
14. J. Wang, W.-J. Yang, and R. Acharya, Colour clustering techniques for colour-content-based image retrieval from image databases, in Proc. IEEE Conf. on Multimedia Computing and Systems, 1997.
15. M. Swain and D. Ballard, Colour indexing, International Journal of Computer Vision 7(1), 1991.
16. M. Ioka, A Method of Defining the Similarity of Images on the Basis of Colour Information, Technical Report RT-0030, IBM Research, Tokyo Research Laboratory, Nov. 1989.
17. W. Niblack, R. Barber, and et al., The QBIC project: Querying images by content using colour, texture and shape, in Proc. SPIE Storage and Retrieval for Image and Video Databases, Feb. 1994.
18. M. Stricker and M. Orengo, Similarity of colour images, in Proc. SPIE Storage and Retrieval for Image and Video Databases, 1995.
19. J. R. Smith and S.-F. Chang, Single colour extraction and image query, in Proc. IEEE Int. Conf. on Image Proc., 1995.
20. J. R. Smith and S.-F. Chang, Tools and techniques for colour image retrieval, in IS & T/SPIE Proceedings, Vol. 2670, Storage & Retrieval for Image and Video Databases IV, 1995.
21. Cao LiHua, Liu Wei, and Li GuoHui, "Research and Implementation of an Image Retrieval Algorithm Based on Multiple Dominant Colours ",Journal of Computer Research & Development, Vol 36, No. 1,pp.96-100,1999.
22. Song Mailing, Li Huan, "An Image Retrieval Technology Based on HSV Colour Space", Computer Knowledge and Technology, No. 3,pp.200-201, 2007.
23. D.C. He and Li Wang. Texture filters based on texture spectrum. Pattern Recognition, 24(12):1187–1195, 1991.

AUTHORS PROFILE

J. Sreedhar obtained my M.Tech(C.S.T) from Andhra University Vizag , Pursuing Ph.D in Computer Science & Engineering discipline from JNT University, Kakinada. Working as Assoc Professor in the Department of CSE, DVRCET, Hyderabad.

Dr. S. Viswanadha Raju obtained his Ph.D in Computer Science & Engineering from ANU. He obtained his M.Tech in CSE from JNTUniversity. He has a good academic background with a very sound and academic research experience. At present he is working as a professor in School of Information Technology in JNTUniversity, Hyderabad. He is guiding 10 research scholors for Ph.D and also conducted several conferences/workshops/seminars with sponsored agencies such as AICTE, DST, TCS, IEEE and CST. His research includes Information Retrieval, Databases, Image Retrieval, Data Mining and related areas. He published 25 research papers in reputed International Journals/Conferences proceedings in his research area. He is active member in different professional bodies with life membership like IETE, ISTE and CSI.

Dr. A. Vinay Babu,. obtained his Ph.D in Computer Science & Engineering from JNTU. He obtained his M.Tech in CSE from JNTUniversity. He obtained his ME from Osmania University. He obtained his BE in ECE from Osmania University He has a good academic background with a very sound and academic research experience. At present he is working as a professor in School of Information Technology in JNTUniversity, Hyderabad. He is guiding 10 research scholors for Ph.D and also conducted several conferences/workshops/seminars with sponsored agencies such as AICTE, DST, TCS, IEEE and CST. His research includes Information Retrieval, Databases, Image Retrieval, Data Mining and related areas. He published 82 research papers in reputed International Journals/Conferences proceedings in his research area. He is active member in different professional bodies with life membership like IETE, ISTE and CSI.