Spatial Color Indexing Using Data Clustering Technique

P J Dutta, D K Bhatatcharyya, M Dutta
Department of Comp Sc & IT, Tezpur University, Napaam, India
{pjd,dkb,malay}@tezu.ernet.in
and
J K Kalita
Dept of Comp. Sc, University of Colorado, Colorado Spring, USA
kalita@pikespeak.uccs.edu

ABSTRACT

This paper presents an efficient spatial indexing technique for content-based image retrieval. Spatial index is generated based upon a fast and robust clustering technique, which can recognize color clusters of any shape. It also exploits entropy measure to decide whether quantization is needed before clustering. Based on experimentation, the performance of the new indexing technique has been found to be better than many of its counterparts providing for robust and compact index, better analysis of the color content of the image, higher precision and recall in retrieval and better retrieval time.

Keywords: Content-based, spatial-index, clustering, object, entropy, precision, recall

1. INTRODUCTION

With the explosive advancement in imaging technologies, image retrieval has attracted increasing interest of researchers in the field of digital libraries, image processing and database systems. Motivated by the ultimate goal of automatically computing efficient and effective descriptors which symbolize various properties of images, recent research on image retrieval systems has been directed towards the development of content-based retrieval techniques for management of visual information such as color, texture, shape and spatial constraints ([1]-[5]). As color plays an important role in image composition, many color indexing techniques have been studied. Out of them, color histogram is one of the most important techniques for content-based image retrieval [1] because of its efficiency and effectiveness. However, due to the statistical nature, a color histogram can only index the color content of images in a limited way i.e., it does not include spatial information for colors in the image. To make color histogram more effective for image indexing, spatial information is also essential ([3],[6],[7],[8]).

Any colored image can be viewed as a distribution of colored pixels in a two dimensional plane. This distribution of colored pixels forms clusters of arbitrary shape within the image. These clusters may have sharp edges demarcating a boundary between changes of color distribution or have diffused or fuzzy boundaries. Hence, the image pixels can be viewed as a data set having two dimensions viz., color and position. Now, considering the color content and the positional aspects of each pixel, an image can be characterized by a set of objects of interest referred to as color clusters of arbitrary shapes. Once the color clusters, henceforth referred to as objects, are identified, it becomes easier for global or regional similarity search of images. In this paper, a new spatial color indexing scheme for content-based retrieval is introduced which is designed based on a robust color clustering technique in a two-dimensional plane. The indexing technique can be found to be significant due to its robustness, compactness, and higher precision & recall.

2. RELATED WORKS

There are several techniques proposed to integrate spatial information with color histograms. Hsu et al. [3] integrate spatial information with color histograms by first selecting a set of representative colors and then analyzing the spatial information of the selected colors using maximum entropy quantization with event covering method. Stricker and Dimai [2] partition an image into five partially overlapping, fuzzy regions, extract the first three moments of the color distribution of each region, and then organize them into a feature vector of small dimension. Smith and Chang [7] apply back-projection on binary color set to extract color regions. Pass and Zabih [9] define the concept of color coherent vector (CCV) and use it to split a color histogram vector into two parts: a coherent vector and a non-coherent vector. A pixel is coherent if it is a part of some sizable similar-colored region, i.e., if its connected component is large enough and non-coherent otherwise. A color coherent vector of an image is the histogram over all coherent pixels of the image. Later, Huang [4] proposes color correlogram for refining histograms. A correlogram is a table indexed by color pairs and distance, where k-th entry for <i,j> specifies the probability of finding a pixel of color j at a distance k from a pixel of color i. A Rao et al. [8] have used annular, angular and hybrid histograms for indexing spatial distribution density of colors. The two dimensional image is uniformly partitioned into N concentric circles with origin at the centroid of the image to form N concentric circles. Similarly, the image is uniformly subdivided into N fan-like sectors with the centroid of the image as the origin. Annular and Angular histogram is defined by the distribution of the pixels in each of the sectors. By combining the above two approaches, the Hybrid histogram is defined.

![Figure 1. Architecture of the proposed CBIR system](image)

Though annular, angular and hybrid histogram methods improve performance over the traditional histogram and the color coherent vector method, they lack in identifying...
an object of interest within the image. Also the index generated is of higher dimension affecting retrieval performance. This paper presents a new spatial color indexing scheme, which differs from the approaches reported so far, in the following way:

- We believe that quantization is not a must for all classes of images; it may cause deterioration in the quality of the index generated. Based on observation, we find that in the average cases, the color content of the images is limited to a few colors only. Thus, in the proposed approach, the necessity of color quantization is initially judged by calculating the entropy value of the color contents.
- We exploit spatial as well as color content information in the identification of the objects of interest in an image, in a realistic manner by applying a density based spatial color clustering algorithm. A colored image, which is a distribution of colors, may contain some noisy distribution of colors that does not impart any meaning to the original picture. Once the objects of interest are identified, it also eliminates the noisy distribution of colors from the index by applying the appropriate method.

Next, we present our scheme.

3. THE PROPOSED SPATIAL INDEXING SCHEME

Our indexing scheme works in four steps. In step 1, it accepts the input query image and computes the entropy value for it. Step 2 checks the entropy value to decide whether quantization on the input image is needed or not. If needed, the quantization module is invoked. In step 3, a special density-based clustering technique is applied to identify the objects of interests. In step 4, the proposed content-based index is generated for each identified object of interest by extracting its four salient features. Finally, the query results are given based on the inferences made by a matching engine. The architecture of the proposed scheme is depicted in Figure 1. Next, we describe each of the modules present in the architecture, in brief.

Entropy Calculation
In most of the CBIRs, the input image is first quantized, to reduce the dimensionality of the index. However, with the dimensionality reduction, a common disadvantage is that it may deteriorate the quality of the image and hence, may cause many false hits. Based on experimentation, it has been found that in the average case, a colored image contains few colors and hence quantization in such cases may be inappropriate. Considering this, by exploiting the entropy concept, an exhaustive experimentation was carried out on a large collection of images (around 10,000 images, downloaded from the Web). The entropy is a measure of the information of the color content of the image [5]. It can be calculated for a colored image as: \( H(v) = - \sum v_i \log(v_i) \) where \( v_i \) is the percentage of pixels in the image which belongs to color \( i \). \( H(v) = 0 \) implies that all the pixels of the image are of same color. \( H(v) \) is maximized when all possible colors in the color space of the image are equally represented. Based on experimentation, it can be found that the average entropy is nearly 2, which indicates that the images whose entropy is more than 2 contains more colors and needs quantization.

Color Space and Quantization
A color space is used to specify a three-dimensional color co-ordinate system and a subspace of the system in which colors are represented as points. The most common color space for digital images and computer graphics is the RGB color space in which colors are represented as a linear combination of red, green, and blue color channels. The properties that are most important in color space processing are uniformity, completeness, and uniqueness. The RGB color space is not perceptually uniform. The distance between two points in the color space does not suggest that the two colors are similar or dissimilar. Additionally, the three-color channels of the RGB color space do not vary consistently with one another with respect to brightness. Therefore, the pixels of the images in the image database and query examples must be transferred in to an alternative color space that satisfies the three properties.

Other color spaces such as CIE-LAB, CIE-LUV and Munsell offer improved perceptual uniformity [12,7]. In general they represent with equal emphasis three variants that characterize color: hue, value and saturation. Hue is that attribute of a color by which we distinguish colors. Value indicates the lightness of a color and Saturation indicates the density of color pigments i.e., amount of color present.

Quantization of the color space is necessary to reduce the dimensionality of the index that characterizes an image at the cost of the quality of the index. The proposed scheme quantizes the color contents of the query image over HSV color space. It basically attempts to quantize the hue component of each color pixel by balancing the visual fidelity and the dimensionality of the resulting quantization. The Human Visual System (HVS) discerns the changes in the hue component by much smaller gaps than changes in the saturation and value. Based on experimentation it has been observed that partitioning the buckets with an equidistant interval of 5 is more justified. Thus the total number of buckets over the hue-axis is 20. The images that have entropy value greater than 2 are quantized with this partition.

Spatial Data Clustering Using Density/Neighborhood Approach
Considering the color content and the positional aspects of each pixel, an image can be characterized by a set of color clusters referred to as objects of interest of arbitrary shapes. Hence to identify the color clusters in an image, any density-based clustering technique can be applied.

In density based clustering, a cluster is defined as a connected dense component, growing in any direction where density leads. Several useful clustering techniques have been proposed based on this concept. However, none of these can be found to operate in linear time. We call our clustering technique BOO-Clustering, which is a variant of a color segmentation technique, operating in linear time. It accepts either the original or the quantized image as input and creates clusters by expanding each core object based on the population count in its neighborhood.

Background of BOO-Clustering Technique
The proposed clustering technique is designed based on the following definitions:

Definition 1: Unclassified Pixels are those pixels, which are not yet clustered. Classified pixels are those pixels, which belong to a particular color cluster.

Definition 2: Target Pixel (t) is a pixel, which is in hand to be classified.
**Definition 3:** A template \( (T) \) of a target pixel \( t \) at the position \( (x, y) \) is a set of pixels which are already classified and are at a neighborhood distance \( d \) from \( t \). The mathematical model is defined in the next section. A sample template \( (T) \) is shown in Figure 2.

**Definition 4:** Pixel Threshold \( (PT) \) is the minimum number of classified pixels required in a cluster so that it does not become a Noisy Cluster. Experimentally, it has been found that a cluster having fewer than the number of pixels in the template does not carry useful meaning of the image. Hence, these noisy clusters are discarded.

![Figure 2: A Sample Template Shown in an Example Image.](image)

The BOO-Clustering Technique

Figure 2 depicts an analogous figure to an image where each block represents a pixel. Here we have not shown colors in each block but some blocks are marked as classified (C), some are marked as unclassified (U), some are marked as pixels of a template \( (TC) \) and one block as target pixel \( (T) \). The process of clustering starts from the position \((0,0)\) of the template (i.e., upper left corner) and successively proceeds till the last pixel is classified. After completion of the clustering process, each pixel is assigned with a Cluster Id and thus with the help of this Cluster Id, the color clusters of the image could be identified. The concept behind BOO-Clustering to assign a Cluster Id to a target pixel \( (t) \) is that it searches in the template \( (T) \) of the target pixel for similar colored pixels. The template \( (T) \) of the target pixel \( (t) \) is constituted of a set of pixels \( (TC) \), which are already classified i.e., pixels which have already been assigned with a Cluster Id. Three cases may arise after the search operation. First, not a single pixel is found which is of the same color as that of the target pixel. In that case, a new Cluster Id is assigned to the target pixel \( (t) \). It implies that there is not a single cluster having the same color as that of the target pixel nearby. Second, if some or all pixels have the same color as that of the target pixel and those pixels have a common Cluster Id, then it assigns that common Cluster Id to the target pixel \( (t) \). This implies that there is a single cluster having the same color as that of the target pixel and they have more than one different Cluster Ids, implies that there is more than one cluster having the same color as that of the target pixel. But, as they are appearing in the same template of the target pixel, they should belong to the same cluster. Hence, in such a situation the algorithm assigns any one of the found Cluster Ids to the target pixel and merges those found color clusters of the template having different Cluster Ids to one cluster. While merging the algorithm traverse back re-clustering successively pixel by pixel till it reaches the first pixel or does not find any one of those clusters in the template of the back tracking pixels. After assigning a Cluster Id (new or already assigned) to a target pixel, the algorithm takes up the next pixel as target pixel and starts the process of clustering. The BOO-Clustering algorithm proceeds sequentially from the first pixel till it encounters the last pixel. In the step of execution, if the third case arises, the algorithm backtracks to the first pixel for merging of clusters (re-clustering) and then comes back to the position where it has left and starts clustering from the immediate next pixel. This is the heart of the algorithm which works like a BOOMERANG. Hence the name BOO-Clustering. But, this algorithm is of order \( N^2 \). To make it linear, instead of backtracking, a link is maintained in a graph for those Cluster Ids that are found more than one in the template of a target pixel.

**Data Structure/Symbols Used:**

- \( \text{Image}(m, n) \): array to hold spatial color data of an input color image where \( m \) is row & \( n \) is column of the array.
- \( \text{Cluster}(m, n) \): array to hold the cluster id of the relative pixel of the \( \text{Image} \).
- \( d \): Number that defines the size of the \( \text{Template} \).
- \( \text{Template} \ (T) \) : \( T = \{(x+i, y+j) \mid \text{if } x+i \ge 0 \text{ and } y+j \ge 0 (i=1,2,3,..; j=0,1,2,...; d) \}

\( P((x+i, y+j)) \): \( \text{if } x+i \le m \text{ and } y+j \le n (i=0,1,2,...; j=1,2,3,...; d) \) for a \( \text{Target_pixel} \).  

**Algorithm BOO-Clustering(Image(m,n))**

for \( \text{row} = 1 \) to \( m \)
for \( \text{col} = 1 \) to \( n \)
    if (\( \text{row} == 1 \) and (\( \text{col} == 1 \)) \( \text{Cluster}(\text{row}, \text{col}) = 1 \);
    else \( \text{Search_in_template}(\text{row}, \text{col}, \text{Image(row,col)}) \);
endfor
endfor

**Search_in_template(row, col, Target_pixel)**

Search in \( \text{Template} \) for Pixels having the same color value as that of \( \text{Target Pixel} \);
Case 1: Not a single pixel found;
    assign \( \text{cluster}(\text{row}, \text{col}) \) with a new cluster id;
Case 2: Some pixels found and they all have same Cluster id;
    assign \( \text{cluster}(\text{row}, \text{col}) \) with found Cluster id;
Case 3: Some pixels found and they have different Cluster ids;
    assign \( \text{cluster}(\text{row}, \text{col}) \) with any one found Cluster ids;
    \( \text{generate_graph}(\text{row}, \text{col}, \text{list of cluster ids}) \);

**generate_graph(row, col, list)**

generate adjacency matrix with list of cluster_no;
The size of the image is \( m \times n = N \) number of pixels and \( d \) is a number which defines the size of the neighborhood template \( T \) for an unclassified target pixel. Figure 2 depicts a template for an unclassified target pixel \( (i) \) where \( d = 3 \). Thus the total no of pixels in the template having \( d = 3 \) is \( 2(d^2 + d) = 24 \). The algorithm runs for \( N \) number of pixels. To assign a Cluster ID to each pixel, it searches for similar colored pixels in its neighborhood template at \( 2(d^2 + d) \) times. If pixels having more than one different Cluster Id is found, then generate graph is called to keep a link between the clusters. Hence the time complexity of the algorithm is \( N + 2(d^2 + d) \). Here \( d \) is very small as compared to \( N \) and hence can be neglected. Thus the complexity of BOO-Clustering is \( O(N) \). The proposed clustering technique can be found to be advantageous in comparison to its other counterparts ([10]), in view of the following points: (i) it operates in linear time, (ii) it can identify all shapes (concave as well as convex) of clusters. (iii) it can successfully handle overlapping sparse distribution of colors, i.e., fuzzy distribution, thus it analyses the color content of the image more efficiently than its other counterparts [8] and (iv) it is based on a simplified data structure which makes the scheme efficient in storage as well as execution time. Next, based on the clustering results, and by considering the other important spatial as well as shape features of each cluster, the index of the image is generated.

### The Spatial Object Index

The output of the BOO-Clustering algorithm is a set of objects based on positional distribution of colors. They can have sharp or diffused boundaries depending on the type of image. These objects need to be indexed by some parameters for similarity search. Each object consists of a set of pixels denoted by \( S_q \) of color \( C_q \) that defines an area in the image having sharp or diffused boundary. This object can be represented by its area, perimeter, color, centroid, principal angle formed at the centroid by each pixel of the object. Out of these the perimeter and area are discarded as these two parameters are not translation invariant. The spatial object index of the image consists of a set of objects where each object is represented by the quadruple parameters \(<\text{color}, \text{centroid}, \text{principal direction}, \text{aspect ratio}>\). Here, color represents the hue component of the set of pixels \( S_q \) of the object. Centroid \( (C^q = (x^q, y^q)) \) is a point of an object that may lie within or outside the object and is calculated by taking the mean of all the pixel positions of the set of pixels \( S_q \) of the object.

\[
X^q = \frac{1}{|S_q|} \sum_{(x,y) \in S_q} x;
\]
\[
Y^q = \frac{1}{|S_q|} \sum_{(x,y) \in S_q} y
\]

The principal direction of an object is the average direction or average principal angle of each of the pixels in the object [8] with respect to the centroid \( C^q \). For each pixel of the object, the direction (i.e., principal angle) \( \Theta(x,y) \) of the pixel in the associated coordinate system is calculated via

\[
\Theta(x,y) = \arctan\left(\frac{Y^q - y^q}{X^q - x^q}\right) \pm \pi
\]

where, + and - are to be selected depending on which quadrant the pixel belongs to, and \((x^q, y^q)\) is the coordinate of the centroid of the object. While calculating the principal angle, the original coordinate system is shifted to the centroid of the object. The average direction/principal direction denoted as \( \Theta(S_q) \), is calculated as

\[
\Theta(S_q) = \frac{1}{|S_q|} \sum_{(x,y) \in S_q} \Theta(x,y)
\]

The principal direction of an object is both translation and rotation invariant with respect to the original image [8]. Aspect ratio is the aspect ratio of the image.

### Robustness & Compactness of the Index:

To establish an index to be robust, it has to be robust subject to the three basic transformations, i.e., translation, rotation and scaling. The proposed 4-dimensional index also can be established to be robust in this regard. The first two features i.e. color and centroid will remain invariant subject to any of those three basic transformations because the clustering results produced by BOO-clustering remains the same even if the images are transformed. Now, since color is the average color of the cluster and centroid is the mean of all the pixel positions of the cluster, so it will also remain invariant subject to those transformations. However, subject to non-uniform scaling it may lead to a differed cluster shape, hence a different centroid. In case of the third feature i.e. principal direction it has already been established [8] that it remains invariant subject to translation, rotation and uniform scaling. Finally, in case of the fourth index i.e. aspect ratio also it can be easily proved that it will remain invariant subject to those aforesaid transformations. Thus, the proposed spatial index is robust subject to these three basic transformations. In case of compactness, while comparing with the dimensionality of the feature vector used in Annullar, Angular and Hybrid Histogram [8], i.e. 2048, a huge number, the proposed method uses only four features. Thus the proposed index is quite compact than its other counterparts [8].

### Database Organization

The spatial object indices of the image are stored in a spatial object tree, which is a variant of B-tree (Figure 3). The root node of the tree points to \( k \) independent tree structures, where \( k \) is the number of parameters in the index. If a new parameter is to be accommodated (i.e., for a \( k+1 \) dimensional index), the root node has to be updated by insertion of a new pointer and accordingly, an associated tree structure will have to be generated. Each of the parameter trees will maintain the parameter key value.
list of Image IDs (i.e., PIDs). For an image, the spatial index which, is a set of objects, will have the same ImageID in many image ID List of different objects of the spatial index.

Matching Engine
The matching engine first evaluates the spatial object index of the query image. It has the facility of searching images in the database based on the parameters of interest and also on objects of interest. Both the conjunctions AND and OR can be used both on objects and parameters. For example, for a query image having five objects \((O_1, O_2, O_3, O_4, O_5)\), the query can be, “Search for all images having Color value AND Centroid value OR Aspect Ratio value for objects \(O_1, O_2\), and \(O_5\) OR \(O_3\)”. For this query, the matching engine starts searching from the root node and will find all images that satisfy the parameter constraints, i.e., have the same Color value AND Centroid value OR Aspect Ratio value for the object \(O_1\). Similarly, the matching engine will fetch images for \(O_2\) and \(O_3\). Finally, these sets of images will be merged together according to the constraints \(O_1\) AND \(O_2\) OR \(O_3\). The merged sets of images will be the result of the given query.

Data Structures and Symbols Used

**Spatial_Index\((O, P)\)**: Spatial Index is a two dimensional array of the Query Image, where \(O\) represents the number of objects in the index and \(P\) represents the number of parameters of the objects.

**Parameter\((4)\)**: Array to hold input parameters (Color, Centroid, Principal Direction and Aspect Ratio) on which search is to be performed.

**PNO** : Total number of parameters in Parameter\((4)\).

**Conj_Parameter\((3)\)**: Array to hold conjunction (AND/OR) for the input parameters.

**Conj_Object\((O-1)\)** : Array to hold conjunction (AND/OR) for the objects in the spatial index.

**Retrieved_Parameter_Image_List** : Linked list to store retrieved Image Ids based on parameter.

**Retrieved_Object_Image_List** : Linked list to store retrieved Image Ids based on objects.

**Matching_engine\((Spatial_index, Parameter, Conj_Parameter, Conj_Object)\)**

Initialize Retrieved_Object_Image_List.

For \(i = 1\) to \(O\)

Initialize Retrieved_Parameter_Image_List.

For \(j = 1\) to \(PNO\)

go to root node

Select pointer contained in Parameter\((j)\)

Search in the selected tree for the corresponding Spatial_Index\((i, j)\) value

\(\text{if found}()\)

\(\text{if } j == 1 \text{ Copy list of Image Ids from the Spatial_Tree node to Retrieved_parameter_Image_List.}\)

\(\text{else Merge list of Image Ids from Spatial_Tree node to Retrieved_Parameter_Image_List with associated conjunction in Conjuction_Parameter\((j-1)\).}\)

\(\text{if } i == 1 \text{ Copy Retrieved_Parameter_Image_List to Retrieved_Object_Image_List.}\)

\(\text{else Merge Retrieved_Parameter_Image_list to Retrieved_Object_Image_List with associated conjunction in Conjuction_Object\((i-1)\).}\)

Efficiency of Retrieval
There are two phases to the computation involved in querying an image database. First, calculation of index of the query image and second, comparison of the generated index with the stored indices of the images in the database and subsequently retrieval of the similar images from the image database. The time taken by the proposed method to calculate the index of a query image consisting of \(N\) pixels is \(3\times N\) (Time taken for Entropy calculation = \(N\), for Quantization = \(N\) and for Clustering and Index generation = \(N\)). Time taken by each of the three counterparts [8] to calculate the index is \(N\). The dimensionality of the index of the proposed method is 4 and each of its counterparts is 2048. Hence the comparison time for \(Q\) number of images are \(4\times Q\) and \(2048\times Q\) by the proposed method and each of the three counterparts respectively. Thus the total time complexity of the proposed method is \((3\times N + 4\times Q)\) and that of each of the counterparts is \((N + 2048\times Q)\). Hence the efficiency of the retrieval system basically depends on the comparison part (i.e., \(4\times Q\) and \(2048\times Q\)) then the calculation of the index. From this it can be revealed that as the image database grows in size, the performance of the annular, angular and hybrid histogram falls and that of the proposed method increases.

4. EXPERIMENTAL RESULTS

To test the technique, we considered a downloaded database consisting of 10000 real world and synthetic images divided into 100 similar groups such as scenery, animals, cars, flowers, space, etc. Implementation was carried out for Annular, Angular and Hybrid histogram techniques along with the proposed technique using the BOO-Clustering method in the HSV color space. As the images are not uniform in brightness and saturation, the Hue component of the color has been used for generation of index in all the methods. The proposed method has been implemented in Java. For any input image, the spatial object index is generated, and similarity search is done on a particular object or combination of objects of the image with the help of the object tree. The system also has the provision for searching on a particular or combination of the four parameters of each object in the index of an image. The cosine measure is used as the similarity measure for the parameters color, principal direction and aspect ratio. If the cosine measure falls within the range .8 to 1, the images are said to be similar with the query. For the parameter centroid, a matching template \(MT\) was defined with distance equal to 3. Any point that lies within this template is said to be matching with the query centroid. Figure 5 reflects one of the average query results out of 400 queries, taken at random over the downloaded database.

Performance Comparison
In our experiment, the average precision recall (APR) was calculated for 400 queries. The results are shown in Figure 5. Numerically, the area [11] enclosed by an APR curve
and the axes as a performance metric, called performance area, is defined as
\[
\frac{1}{2} \sum_{i=1}^{N-1} (x_{i+1} - x_i)(y_{i+1} + y_i)
\]
where \((x_i, y_i)\) is the (recall, precision) pair when the number of retrieved images is \(i\) and \(N\) is the total number of top matches. The proposed method was compared with three popular retrieval techniques [8] i.e., Annular, Angular and Hybrid Histograms (Figure 4). Based on experimental results, it can be found that the performance areas for the curves are: 1735.75 (BOO-Clustering), 1326.45 (Annular), 1278.29 (Angular) and 1186.21 (Hybrid). Hence the improvement produced by the BOO-Clustering method over the other three methods are: 30.85% (Annular), 35.79% (Angular) and 46.33% (Hybrid).

![Figure 4. Comparison of the Hybrid, Angular & Annular histogram with the Proposed](image)

5. CONCLUSION

A better content-based indexing scheme has been presented in this paper. The scheme generates a compact, storage-efficient 4-dimensional transformation invariant index for any color image by successfully utilizing the entropy concept and a robust data clustering technique. The proposed scheme can be found to be superior in comparison to its other counterparts [8].

6. REFERENCES