CHAPTER - 3

Multistage Content Based Image Retrieval

3.1. Introduction

Content Based Image Retrieval (CBIR) is process of searching similar images from the database based on their visual content. A general purpose CBIR has found its applications in many areas. There are many issues which affect the designing of CBIR like selected set of image features, dimension of feature vector, retrieval algorithm and method for presenting final results [Vassilieva (2009)]. Generally CBIR systems use low level features of the image to index and retrieve images from the database. However it is very difficult to fill the semantic gap by using only low level features of an image because of large diversity of image databases. Therefore image retrieval algorithm and method for presenting final results need to be improved so that image features can better represent the semantics of images.

Retrieval algorithms used in traditional CBIR systems search the whole database independently using different image features. Each of them is represented by a point in the corresponding feature space. Some systems use several feature spaces to represent the same feature in order to improve retrieval accuracy. In this case, search in each feature space is also performed independently, followed by data fusion method to merge the retrieved sets (intermediate outputs) into one common output. An output here is a ranked set of retrieved objects, which is an answer of the retrieval system to a given query. To merge the results of retrieval in different feature spaces, it is common to use linear combinations of ranks of an element in each intermediate output as its rank in the common output [Vassilieva (2009)]. For example, if CBIR is based on color, texture and shape features of images, the
system produces intermediate output by comparing color, texture and shape feature respectively with whole database. Fusion and ranking techniques are then employed to merge these intermediate results to produce final output of the system. A detailed description of fusion and ranking techniques used in image retrieval is discussed in [Park et al. (2005)].

These approaches however tend to have less accuracy as all intermediate results are formed by searching the whole database independently based on a specific feature. This approach also takes high computation time as it requires searching of database multiple times and further fusion and ranking of the intermediate results. Apart from this, it is also not efficient to compare the combined feature vector of different feature spaces at once and produce the final result without producing intermediate results. Since different feature spaces contain different values of feature and each feature has different relative importance for retrieval.

To deal with aforementioned issues, a new scheme for image retrieval is proposed in this chapter. To index images in database, global features based on color and texture are computed. These features are combined with contour based shape feature to form a single feature vector to be indexed in the database. When a query image is presented to the system, the retrieval of similar images occurs in stages based on color, texture and shape similarity respectively. The intermediate results thus produced act as an input to the next stage i.e. the output images of each stage act as database for next stage thereby reducing the number of images to be compared at each stage. This approach also eliminates the requirement of fusion and normalization techniques required for finding final similarity score.

An image database contains a wide variety of images but images which are relevant to the query may be few. To better meet the user’s intent, the proposed system performs search in relevant images only. This is different from traditional CBIR systems which searches the whole database for every
feature. Relevance of the images is first established by comparing their color feature. Search based on texture and shape features is performed only on the images which have color similarity with query image. This approach reduces the diversity of database by removing irrelevant images at each stage so that low level features can better represent the semantic of images. Experiments have shown that the system produced the desired results with greater accuracy.

3.2. Related Work

Traditional CBIR systems search independently in each feature space under consideration and use fusion and ranking techniques to merge intermediate results to produce final similarity score. Two commonly used fusion techniques are weighted sum of individual distances and linear combination of individual distances in a sorted order. We have studied image matching techniques employed in different benchmark image retrieval systems. Some of the significant techniques are discussed in this section.

In Compass [Brunelli (2000)], the distance between two images is a weighted sum of the individual feature distances. The cloud of query feature vector in feature space is clustered into a number of query sets.

Visual Seek [Smith and Chang (1997)], finds the matches of a query image with a single region, queries on color set, region absolute location, area and spatial extent independently. The results of these queries are intersected and from the obtained candidate set, the best matching images are taken by minimizing a total distance given by the weighted sum of the four distances mentioned.

Draw search [Sciascio et al. (1999)], uses color/shape subsystem; the similarity between two feature vectors is given by the cosine metric. The similarity score between a query and a
A database image is calculated as a weighted sum of the distances between the two color vectors and shape descriptors.

In FIDS (Flexible Image Database System) [Berman and Shapiro (1999)], the distance between wavelet coefficients, is some weighted difference. An overall distance can be calculated by taking the weighted sum, maximum, or minimum of the individual feature distances, which preserves metric properties.

In VIR Image Engine [Bach et al. (1996)], a similarity score is computed for each primitive in the current query combination using the distance function defined within the primitive. These individual scores are combined in an overall score using a set of weights in a way characteristic to the application. This score is then stored in a score structure, which also contains the individual similarity scores for each primitive. This allows a quick re-computation of the overall score for a new set of weights.

In TODAI (Typographic Ornament Database and Identification) [Michel et al. (1996)], the similarity between two images is calculated as the sum of the Euclidean distances of the six feature vectors.

Wang et al. (2011) have proposed an image retrieval scheme combining color feature like dominant color of region, texture feature like steerable filter and shape feature based on pseudo Zernike moment. For calculating similarity between each feature different similarity measure are employed. Final similarity between query image (I) and database image (Q) is calculated by taking the weighted sum of individual feature distances given as:

$$S(I, Q) = W_C S_{\text{Color}}(Q, I) + W_T S_{\text{Texture}}(Q, I) + W_S S_{\text{Shape}}(Q, I),$$

(3.1)

where, $S_{\text{Color}}(Q, I)$, $S_{\text{Texture}}(Q, I)$ and $S_{\text{Shape}}(Q, I)$ are individual distances of color, texture and
shape feature respectively. $w_c$, $w_t$ and $w_s$ are the weights. The retrieval model of a typical CBIR system based on color, texture and shape features is shown in Figure 3.1. Experimental results show the efficacy of the method.

![Figure 3.1: Wang et al. (2011) model of image retrieval](image)

In Blobworld [Carson et al.(2002)], the quadratic form distance is used to match two color histograms. The distance between two texture descriptors is the Euclidean distance between their coordinates in representation space. The distance between centroids is the Euclidean distance. The distances are combined into a single final distance.

In KIWI (Key-points Indexing Web Interface) [Loupias and Bres (2001)], the color feature vectors are compared with the Euclidean distance and 2D histograms are compared using the Bhattacharyya distance. After a normalization of the distribution of distance values from the individual features, the similarity values are sorted and then weighted averaged.
In Metaseek [Benitez et al. (1998)], color and texture are extracted locally by MetaSEEk for the clustering. Color similarity is computed by calculating the color histogram of an image. Texture is computed by measuring the coarseness, contrast, and presence/absence of directionality of an image. The distance between two feature vectors is the Euclidean distance.

In PicHunter [Cox et al. (2000)] the distance between individual features (color vectors or annotation lists represented as binary vectors) is the $L_1$ distance. These distances are scaled and combined in a global distance. The scaling factors are computed by maximizing the probability of a training set.

Lu and Chang (2007) used two different measurements for the global features and the local features to evaluate the similarity between the two images. For the global color features, the scheme uses Euclidean distance to calculate the similarity. On the other hand, for the local feature, the scheme uses hamming distance to evaluate the distance between the two bitmaps. Afterwards, the overall similarity is obtained by linearly combining these two similarity values. However, the linear combination will become meaningless because the magnitude similarity value may dominate the others therefore Gaussian normalization is used to normalize the features into the same criterion.

All these techniques require searching the whole database multiple times, depending upon the number of feature spaces used for retrieval. Also these intermediate search results require fusion and ranking techniques to produce final similarity result. Normalization of features is also required since the magnitude similarity value may dominate the others. Also appropriate value of weights must be assigned to get good results. This process involves complex calculations and lot of computation time. In addition to this, accuracy of the system is also affected because the
search encompasses the whole database which contains a wide variety of images. The proposed technique tries to overcome these issues while increasing the accuracy of the system.

3.3 Multistage CBIR

The architecture of the proposed Multistage CBIR system is discussed in the following subsections:

3.3.1 Image Indexing

Images to be stored in database are taken one by one and their color, texture and shape features are calculated as shown in Figure 3.2.

![Figure 3.2: Content Based Image Indexing](image)

For each image color feature is extracted using quantized color Histogram in HSV color space. Number of pixels in each bin of histogram is used to form a color feature vector. To get the texture information, Gabor texture features are computed and stored in the database. Similarly, shape feature vector is constructed by computing Fourier descriptor based on centroid distance. All three feature vectors are combined to form a single feature vector which is
indexed in the database using primary key as index.

The calculated feature vector is stored in the relational database table using primary key as sequential counter (see Table 3.1). The initial value of the counter is set as 1. The value of counter is increased by one automatically, when an image feature vector is stored into the database.

Table 3.1: Structure of database

<table>
<thead>
<tr>
<th>Index</th>
<th>Color Feature</th>
<th>Texture Feature</th>
<th>Shape Feature</th>
<th>Image Path/Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C₁</td>
<td>C₂</td>
<td>Cₙ</td>
<td>Tₙ₊₁</td>
</tr>
<tr>
<td>1</td>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>2</td>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
</tbody>
</table>

The name of images in the database are unique i.e no two images in the database can have the same name. The color feature is represented by its components (C₁, C₂...Cₙ). Similarly texture and shape features are also described using their components.

3.3.2 Image Retrieval

The proposed image retrieval algorithm is very simple but is quite effective. The process of querying and retrieving similar images from database is divided into three stages, as shown in Figure 3.3. After each retrieval steps, the proposed method find the indexes of the retrieved image.

In the first stage, color feature vector of the example image is calculated, as done in the
indexing phase. This feature vector is compared only with color feature of all other images (M) in the database using histogram intersection distance (S_1). The output images are ranked in increasing order of their distance with the query image. Top N (<M) images which are closer to the query image are retrieved and presented as output of I stage as intermediate result R_C. Indexes of these N images are stored in an array. This array is then used to query the database to retrieve texture feature in a sequential manner by using the index as variable in SQL query. Retrieved set of texture features is stored in a cell array variable and converted into the format as returned by the database cursor. This process reduces the database virtually to N images having the only texture feature. Thereby removing irrelevant images and narrowing down the range of images to be searched in the next stage.

In the second stage, Gabor texture feature of the example image is computed. This texture feature vector is compared only with the texture feature vectors of output images of I stage (i.e. only N images) using Euclidean distance (S_2). Top P (< N) images which are closer to query image in terms of texture are presented as output (R_T) of stage II. Indexes of these P images are retrieved by querying the database using the path of retrieved images. These indexes are stored in an array and are used to make the database for shape feature as done for texture feature in the first stage.

In the third stage, shape feature vector of the example image is extracted and is compared with the shape feature of all the images which were output of the second stage (i.e. only P images). Top K (< P) images which are most similar to query image are the final output of the system.

Comparing images in this way reduces the number of irrelevant images at each stage, which may alter the accuracy of output. This proposed method of retrieval can produce better results.
Figure 3.3: Proposed model of image retrieval
even with simple set of image feature [Shrivastava and Tyagi (2012)]. The values of \( N, P \) and \( K \) can be set according to need or diversity of the database used. For example for the Corel 1000 image database (\( M=100 \)) value of \( N, P \) and \( K \) can be set as 100, 50 and 20 respectively.

Figure 3.4 shows the relationship between database and different intermediate results of respective stages. Let \( D \) be the universal set representing database containing \( M \) images. \( R_C \) is the set of top \( N \) similar images retrieved from the database by performing search based on color feature.

**Figure 3.4: Venn diagram showing relationship between database (D) and intermediate results \( R_C, R_T \) and \( R_S \)**

Then set \( R_C \) can be represented as:

\[
R_C = \{ x \mid x \in D \} \text{ and } n(R_C) = N \text{ where } N < M \text{ hence } R_C \subseteq D \tag{3.2}
\]

Let \( R_T \) be the set containing similar images in terms of texture. \( R_T \) is obtained by performing search in \( R_C \) by texture features and taking top \( P \) images of the sorted result. Hence the set \( R_T \) can be represented as:

\[
R_T = \{ x \mid x \in R_C \} \text{ and } n(R_T) = P \text{ where } P < N \text{ hence } R_T \subseteq R_C \tag{3.3}
\]

Similarly, \( R_S \) represents the set of images having shape similarity with the query image. \( R_S \) is
formed by comparing images in $R_T$ with the query image in terms of shape feature and taking top $K$ images of the sorted result. $R_S$ can be represented as:

$$R_S = \{ x | x \in R_T \} \text{ and } n(R_T) = K \text{ where } K < P \text{ hence } R_S \subset R_T$$ (3.4)

From equations 3.2, 3.3 and 3.4, the relation between $R_C$, $R_T$, and $R_S$ can be given by:

$$R_S \subset R_T \subset R_C \subset D \text{ since } K < P < N < M$$ (3.5)

$R_S$ is the final output of the system.

3.4. Feature Extraction

For establishing the efficacy of proposed matching technique, most commonly used features for color, texture and shape are employed for indexing images in the database. This section describes the details of feature extraction process and similarity measure used at each stage.

3.4.1 Stage I

Color is most commonly used feature in the CBIR, since it is not affected by rotation, scaling and other transformations on the image. Color features are generally represented by the color histogram. Computation of color histogram requires quantization of selected color space. In this work, we have selected HSV (Hue, Saturation, Value) color space, since it is more perceptually uniform than other color spaces [Swain and Ballard(1991), Gonzalez and Woods (1992)]. In stage I, Global color histogram of query image is compared with pre computed histogram data of all other images in the database using histogram intersection distance. Computation of global color histogram is done using following steps:

**Step 1.** Convert images from RGB to HSV color space.
Step 2. Apply non-uniform quantization technique as given below:

\[
H = \begin{cases} 
0 & h \in [340, 20] \\
1 & h \in [20, 50] \\
2 & h \in [50, 75] \\
3 & h \in [75, 140] \\
4 & h \in [140, 160] \\
5 & h \in [160, 195] \\
6 & h \in [195, 285] \\
7 & h \in [285, 305] \\
8 & h \in [305, 340] 
\end{cases}
\]

\[
S = \begin{cases} 
0 & s \in [0, 0.2] \\
1 & s \in [0.2, 0.65] \\
2 & s \in [0.65, 1] 
\end{cases}
\]

\[
V = \begin{cases} 
0 & v \in [0, 0.2] \\
1 & v \in [0.2, 0.7] \\
2 & v \in [0.7, 1] 
\end{cases}
\]

Step 3. Plot HSV color histogram of 81 bins.

Step 4. Save each bin value in database to form a color feature vector.

Step 5. Compute similarity using histogram intersection distance using:

\[
d(h, g) = \frac{\sum \sum \sum \min(h(a, b, c), g(a, b, c))}{\min(|h|, |g|)},
\]

where \(h\) and \(g\) are histogram and \(|h|\) and \(|g|\) gives the magnitude of each histogram, which is equal to the number of samples. Colors not present in the user’s query image, does not contribute to the intersection distance. This reduces the contribution of background colors. The sum is normalized by the histogram with fewest samples.

The output images of stage I are sorted according to their distance with the query image and top \(N\) images of the sorted result called as \(R_C\) are taken as input to the next stage thereby reducing the database images to be compared at each stage.
3.4.2 Stage II

In stage II, Image retrieval is done using Gabor texture feature. Gabor filter (or Gabor wavelet) is widely adopted to extract texture features from the images for image retrieval [Manjunath et al. (2001)], and has been shown to be very efficient. Basically, Gabor filters are a group of wavelets, with each wavelet capturing energy at a specific frequency and a specific direction. Expanding a signal using this basis provides a localized frequency description, therefore capturing local features energy of the signal. Texture features can then be extracted from this group of energy distributions. The scale (frequency) and orientation tunable property of Gabor filter makes it useful especially for texture analysis [Sebe and Lew (2000)]. The procedure for extracting the Gabor feature vector is given as

*Step 1.* For a given image \( I(x, y) \) of size \( P \times Q \), the discrete Gabor wavelet transform is given by a convolution:

\[
G_{mn} = \sum_s \sum_t I(X - s, Y - t) f^*_{mn}(s, t),
\]

(3.8)

where \( s \) and \( t \) specify the filter mask size which is set as 60 \( \times \) 60. The * sign indicates complex conjugate of the mother wavelet. \( m, n \) specify the scale and orientations of wavelet respectively.

*Step 2.* Apply Gabor filter of different orientations at different scales on the transformed image and obtain an array of magnitudes given by equation (3.9).

\[
E(m, n) = \sum_x \sum_y |G_{mn}(X, Y)|,
\]

(3.9)

*Step 3.* Calculate the mean \( \mu_{mn} \) and standard deviation \( \sigma_{mn} \) of magnitudes of the transformed coefficient:
\[ \mu_{mn} = \frac{E(m,n)}{P \times Q}, \quad \text{(3.10)} \]

\[ \sigma_{mn} = \sqrt{\frac{\sum_{x} \sum_{y} (|f_{mn}(x,y)| - \mu_{mn})^2}{P \times Q}}, \quad \text{(3.11)} \]

**Step 4.** A feature vector \( FV \) is created using 5 scales and 6 orientations which is given by:

\[ FV = (\mu_{00}, \sigma_{00}, \mu_{01}, \sigma_{01}, \ldots, \mu_{4}, \sigma_{4}) \]  
\[ \text{(3.12)} \]

**Step 5.** Similarity between the query image feature vector \( Q \) and the target image feature vector \( T \) is given by:

\[ D(Q,T) = \sum_{m} \sum_{n} d_{mn}(Q,T), \quad \text{(3.13)} \]

where,

\[ d_{mn} = \sqrt{(\mu_{mn} - \mu_{mn}^T)^2 + (\sigma_{mn} - \sigma_{mn}^T)^2}, \quad \text{(3.14)} \]

The output images of stage II are sorted according to their distance with the query image and top \( P(<N) \) images of the sorted result called as \( R_T \) are taken as input to the next stage thereby reducing the database images to be compared at next stage.

**3.4.3 Stage III**

In this stage, shape feature based on Fourier descriptor are extracted from the query image and are compared with the corresponding shape feature vectors of \( P \) images which were presented as output of stage II. Fourier descriptors based on complex coordinate and centroid distance have been extensively used in shape based image retrieval. Many authors have shown that centroid distance based Fourier descriptor performs better than other because they are translation, rotation and scaling invariant [Zhang et al. (2001), Mingqiang et al. (2008)]. Since
segmentation of all images cannot be done accurately therefore the object having largest connected boundary is used to calculate the Fourier descriptor with the assumption that largest object contains the important details of image. The procedure used to obtain Fourier descriptor based shape feature is given as follows:

*Step 1.* Convert image from RGB to Grayscale.

*Step 2.* Convert image from Grayscale to Binary.

*Step 3.* Find the boundaries of all the connected regions in the image.

*Step 4.* Separate the boundary coordinates of the largest connected object from the image.

*Step 5.* Apply polygon fitting algorithm to regularize the shape boundary.

*Step 6.* Calculate 1-D shape signature of the boundary coordinate based on centroid distance which is given by equation (3.15).

\[
 r(t) = \sqrt{([X(t)-X_C]^2 + [Y(t)-Y_C]^2)},
\]

(3.15)

where \((X_C, Y_C)\) are the coordinate of object centroid.

*Step 7.* Calculate Fourier transform of the shape signature as given by equation (3.16).

\[
 a(k) = \frac{1}{N} \sum_{t=0}^{N-1} r(t) \exp(j2\pi k t) / N,
\]

(3.16)

where \(k = -N/2, \ldots, N/2 - 1\)

*Step 8.* Remove the real part of the Fourier coefficient to make it rotation invariant.

*Step 9.* Take values of magnitude of fourteen Fourier coefficients and make them scaling invariant by dividing each of them by the \(FD_1\). Hence the feature vector is given as:
\[ f = \begin{bmatrix} FD_2 & FD_3 & \cdots & FD_{N-1} \\ FD_1 & FD_1 & \cdots & FD_1 \end{bmatrix}, \quad (3.17) \]

**Step 10.** Compare this feature vector with pre-computed shape feature vector of P images of stage II using distance function given by:

\[ d = \sqrt{\sum_{i=0}^{N-1} |f_q^i - f_d^i|^2}, \quad (3.18) \]

where \( f_q^i \) is the \( i \)th component of query feature vector and \( f_d^i \) is the \( i \)th component of feature vector stored in the database, \( d \) is the Euclidean distance between two features. \( N \) is the total number of components or sampled points in the shape contour.

Images are sorted in ascending order with respect to their distances with query image and top \( K (<N) \) images having close similarity with the query image are presented as final output of the system. We denote this output as \( R_s \).

### 3.5 Relevance Feedback

The proposed system can provide better results if the user is aware of the distribution of images in the database. The prior knowledge may help in setting appropriate values of \( N \), \( P \) and \( K \) parameters, which may alter the accuracy of results. However, an effective CBIR should be able to produce better results for novice user also. To help the user to choose appropriate value of \( N \), \( P \) and \( K \), a newer approach for relevance feedback mechanism is implemented in the proposed system which automatically adjusts the values of \( N \), \( P \) and \( K \) based on the number of negative feedbacks given by the user.

Since the accuracy of results largely depends upon the value of \( N \) which creates two disjoint partitions: relevant search area containing \( N \) images and irrelevant area containing \( M-N \) images, where \( M \) is the total number of images in the database. A bad result would mean that a sufficient
number of good images are not coming inside the value of N and hence it should be increased to get better precision value. The amount of increment is determined by the number of negative output produced. The updated value of parameters N and P can be computed as:

\[ N_{\text{new}} = N_{\text{old}} + I_n \]  
\[ P_{\text{new}} = P_{\text{old}} + I_n. \]  

(3.19)  
(3.20)

where, \( I_n \) is the number of negative output images which are irrelevant to the query image. This process of updating the value of N and P continues until value of \( I_n \) becomes 0 or constant in successive iterations. However, if increase in the value of N results in increase in the number of negative outputs \( (I_n) \), then the process of updating is reversed and new values for N and P are calculated by subtracting the value of \( I_n \) from their old value. In the whole process, the parameter K remains fix as it denotes the number of images that the user want in the final output of the system.

### 3.6 Analysis of Computation time

Traditional CBIR systems search all images in the database independently for each feature space being used. Let \( n \) be the total number of images in the dataset and \( f \) denotes the number of feature spaces used for indexing image in the database. Then Computation time \( T \) of the system can be roughly estimated as:

\[ T \geq f \times n, \]  

(3.21)

For example, if the total number of images in the database are 1000 and the system employ three feature spaces for color, texture and shape then the estimated computation time \( T \) will be greater than equal to 3000 (i.e. \( 3 \times 1000 \)) units of time.
In contrast to traditional systems, computation time of the proposed system depends upon the values of parameters N, P and K. The present system searches all images in the database only in the first stage and the number of images reduces successively in later stages depending on the value of N, P and K. The computation time T of the proposed system can be estimated as

\[ T \geq n + N + P, \]  

(3.22)

Now, if \( n = 1000, N=100 \) and \( P=50 \), the overall estimated computation time T will be greater than equal to 1150 (i.e. \( 1000 + 100 + 50 \)) units. In general, the average computation time of the proposed system is 1.5 times lesser than the traditional systems.

### 3.7 Experimental Results

This section provides the experimental evaluation of present method. A computer system having Pentium IV, 2.8 GHz processor and 1 GB RAM is used for conducting experiments. This system has been implemented in MATLAB.

#### 3.7.1 Experiment on Dataset -1

The performance of the proposed multistage retrieval model is tested using Corel database downloaded from [http://wang.ist.psu.edu/docs/related/](http://wang.ist.psu.edu/docs/related/). The corel image gallery contains 1000 natural images having 100 images of 10 categories, which include African people, Beaches, Buildings, Buses, Dinosaurs, Elephant, Flower, Horses, Mountains and Foods respectively. Example images from each of these categories are shown in Figure 3.5.

![Figure 3.5: Example images representing each category of Corel database](image_url)

44
Figure 3.6 shows the example run of our system. Image in the top left corner is the query image and rests of the 20 images are the retrieved result in response to query image. Since values of N, P and K affect the accuracy of our system. In general N should be set equal to the number of images in the database which are relevant to the query image. We have taken these values as 100, 50 and 20. The values of N and P can be set equal for filtering the images based on both color and texture feature.

**Figure 3.6: Retrieval results for an example query**

In addition to this, flexibility of the system is further increased by giving the option of manual setting to the user. Figure 3.7 shows the retrieval results of each stage with value of N, P and K set to 10, 8 and 5 respectively. It is clear that irrelevant images are filtered at each stage thereby narrowing down the search range which makes low level features to better represent the user intent. The proposed system has been tested using default option for calculating the average precision of different category. A retrieved image is said to be relevant if it belongs to the same category as query image.
Present system is evaluated by taking each image in each category as the query image. Retrieval results of each of these images are used to calculate the average precision and average recall for that category with number of output images (L) set as 20.

Evaluation results from proposed system and other five models are shown in Table 3.2. It is observed that our method has better average precision than all other methods in each category except for the category of Buses. However average precision in this category can also be improved by wisely setting the values of N, P and K in our model. Better results are mainly due to refinement of the search process in relevant area of the database.

![Intermediate results for each stage](image)

**Figure 3.7: Intermediate results for each stage**

(a) Result of first stage ($R_C$) with N=10  (b) Result of second stage ($R_T$) with P= 8  (c) Final result of system ($R_S$) with K = 5.

Figures 3.8 and 3.9 show the performance evaluation of proposed model and other models in terms of average precision and average recall respectively with L varying from 20 to 100.
Experimental results reveal that present model is significantly superior to the other models. The results are better as the proposed scheme perform search in only the relevant area of the database.

Table 3.2 Average precision of different models

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Africa</td>
<td>0.748</td>
<td>0.703</td>
<td>0.683</td>
<td>0.720</td>
<td>0.453</td>
<td>0.424</td>
</tr>
<tr>
<td>2</td>
<td>Beaches</td>
<td>0.582</td>
<td>0.561</td>
<td>0.540</td>
<td>0.400</td>
<td>0.398</td>
<td>0.446</td>
</tr>
<tr>
<td>3</td>
<td>Buildings</td>
<td>0.621</td>
<td>0.571</td>
<td>0.562</td>
<td>0.600</td>
<td>0.374</td>
<td>0.411</td>
</tr>
<tr>
<td>4</td>
<td>Buses</td>
<td>0.802</td>
<td>0.876</td>
<td>0.888</td>
<td>0.500</td>
<td>0.741</td>
<td>0.852</td>
</tr>
<tr>
<td>5</td>
<td>Dinosaurs</td>
<td>1.000</td>
<td>0.987</td>
<td>0.992</td>
<td>0.950</td>
<td>0.915</td>
<td>0.587</td>
</tr>
<tr>
<td>6</td>
<td>Elephants</td>
<td>0.751</td>
<td>0.675</td>
<td>0.658</td>
<td>0.600</td>
<td>0.304</td>
<td>0.426</td>
</tr>
<tr>
<td>7</td>
<td>Flowers</td>
<td>0.923</td>
<td>0.914</td>
<td>0.891</td>
<td>0.800</td>
<td>0.852</td>
<td>0.898</td>
</tr>
<tr>
<td>8</td>
<td>Horses</td>
<td>0.896</td>
<td>0.834</td>
<td>0.803</td>
<td>0.630</td>
<td>0.568</td>
<td>0.589</td>
</tr>
<tr>
<td>9</td>
<td>Mountains</td>
<td>0.561</td>
<td>0.536</td>
<td>0.522</td>
<td>0.300</td>
<td>0.293</td>
<td>0.268</td>
</tr>
<tr>
<td>10</td>
<td>Food</td>
<td>0.803</td>
<td>0.741</td>
<td>0.733</td>
<td>0.400</td>
<td>0.369</td>
<td>0.427</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>0.769</td>
<td>0.739</td>
<td>0.727</td>
<td>0.590</td>
<td>0.527</td>
<td>0.533</td>
</tr>
</tbody>
</table>

Figure 3.10 shows the comparison of the average retrieval time for retrieving 20 images for all models. Experimental results show that retrieval time of proposed model on the Corel database of 1000 images is 1.25 sec. which is less than Wang et al. (2011) and is slightly greater than rest
of the four schemes. The reason for this is that the proposed model consumes a lot of time in selecting the images for making the database for intermediate stages. This process involves re-ordering and indexing of images.

![Figure 3.8: Comparison of average precision among different models](image1)

![Figure 3.9: Comparison of average recall among different models](image2)
3.7.2 Experiments on Dataset-2

The second image database used is Cifar-10 that was downloaded from http://www.cs.toronto.edu/~kriz/cifar.html. This images database consists of 60,000 images in 10 classes, 6000 images per class. The class names of these images are: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship and Truck. A few sample of the images in each class is shown in Figure 3.11.

The value of N, P and K are set as 100, 50 and 20. The result of the average precision for top 20 images of the proposed model and the other five models are given in the Table 3.3. It is obvious from the table that the proposed method has significant improvement over the existing methods.

Figure 3.12 shows the average precision and recall of the compared model over randomly selected 100 query images from dataset-2. The results show that the proposed model is significantly better than the other models in comparison.
Figure 3.11: Sample images from each category of the dataset -2

Table 3.3 Comparison of average precision of different models on dataset -2

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Airplane</td>
<td>0.81</td>
<td>0.762</td>
<td>0.732</td>
<td>0.720</td>
<td>0.521</td>
<td>0.568</td>
</tr>
<tr>
<td>2</td>
<td>Automobile</td>
<td>0.762</td>
<td>0.705</td>
<td>0.681</td>
<td>0.400</td>
<td>0.432</td>
<td>0.395</td>
</tr>
<tr>
<td>3</td>
<td>Bird</td>
<td>0.853</td>
<td>0.816</td>
<td>0.789</td>
<td>0.600</td>
<td>0.365</td>
<td>0.464</td>
</tr>
<tr>
<td>4</td>
<td>Cat</td>
<td>0.942</td>
<td>0.921</td>
<td>0.910</td>
<td>0.500</td>
<td>0.861</td>
<td>0.826</td>
</tr>
<tr>
<td>5</td>
<td>Deer</td>
<td>0.936</td>
<td>0.918</td>
<td>0.872</td>
<td>0.950</td>
<td>0.821</td>
<td>0.782</td>
</tr>
<tr>
<td>6</td>
<td>Dog</td>
<td>0.893</td>
<td>0.854</td>
<td>0.756</td>
<td>0.600</td>
<td>0.435</td>
<td>0.482</td>
</tr>
<tr>
<td>7</td>
<td>Frog</td>
<td>0.791</td>
<td>0.768</td>
<td>0.743</td>
<td>0.800</td>
<td>0.692</td>
<td>0.719</td>
</tr>
<tr>
<td>8</td>
<td>Horse</td>
<td>0.921</td>
<td>0.893</td>
<td>0.842</td>
<td>0.630</td>
<td>0.692</td>
<td>0.736</td>
</tr>
<tr>
<td>9</td>
<td>Ship</td>
<td>0.832</td>
<td>0.754</td>
<td>0.682</td>
<td>0.300</td>
<td>0.425</td>
<td>0.568</td>
</tr>
<tr>
<td>10</td>
<td>Truck</td>
<td>0.851</td>
<td>0.826</td>
<td>0.764</td>
<td>0.400</td>
<td>0.642</td>
<td>0.689</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>0.859</td>
<td>0.821</td>
<td>0.777</td>
<td>0.590</td>
<td>0.589</td>
<td>0.622</td>
</tr>
</tbody>
</table>
Figure 3.12: Average precision - recall of different model over randomly selected 100 queries

Figure 3.13: Comparison of average retrieval time among different models for Cifar-10 database of 60000 images

Figure 3.13 gives the comparison of retrieval time on the dataset-2. It is obvious that the proposed model takes significantly less time than all other models. This is because retrieval time of proposed model does not increase much with the increase in the size of the database since our
model compare only a fixed number of images as defined by the system or user. The retrieval
time of the system also varies depending upon the values of N, P and K.

3.8 Conclusion

This chapter presents a new model for retrieving images in stages. The proposed method
works on a three layered feed forward architecture. Each layer narrows down the search range by
filtering irrelevant images based on color, texture and shape features respectively. Retrieving
images in this manner helps in reducing the semantic gap and to an extent eliminate the need of
precise segmentation techniques. Since shape features obtained by segmentation are used to
perform search in only few images. This approach also eliminates the need of fusion and
normalization techniques required to make overall similarity score. The appropriate choice of
weights is also not required for finding final similarity score. Moreover both global and region
features are combined to obtain better retrieval accuracy. Experimental results obtained by
proposed approach are outstanding for most of the query images. The proposed approach has
great potential and provides better flexibility to the user as precision of the system can be
increased to any level by wisely adjusting the value of N, P and K. The present method provides
flexibility of controlling the size of the database according to user interest; this helps in reducing
the diversity of database. Moreover retrieval time of the present approach does not increase
much with the increase in number of images in the database. This makes it suitable for large
image databases. Performance of the proposed system can be improved by employing more
powerful feature set for image representation and by varying the sequence of filtering in
respective stages.