Fast Image Retrieval Using Color-Spatial Information

Beng Chin Ooi *Kian-Lee Tan* *Tat Seng Chua* *Wynne Hsu*

Dept. of Information Systems & Computer Science

National University of Singapore

Lower Kent Ridge Road, Singapore 119260

Abstract

In this paper, we present an image retrieval system that employs both the color and spatial information of images to facilitate the retrieval process. The basic unit used in our technique is a *single-colored cluster*, which bounds a homogeneous region of that color in an image. Two clusters from two images are similar if they are of the same color and overlap in the image space. The number of clusters that can be extracted from an image can be very large, and it affects the accuracy of retrieval. We study the effect of the number of clusters on retrieval effectiveness to determine an appropriate value for "optimal" performance. To facilitate efficient retrieval, we also propose a multi-tier indexing mechanism called the *Sequenced Multi-Attribute Tree* (SMAT). We implemented a two-tier SMAT where the first layer is used to prune away clusters that are of different colors, while the second layer discriminates clusters of different spatial locality. We conducted an experimental study on an image database consisting of 12,000 images. Our results show the effectiveness of the proposed color-spatial approach, and the efficiency of the proposed indexing mechanism.

Keywords: single-colored cluster, content-based retrieval, color-spatial information, sequenced multi-attribute tree.

1 Introduction

With the advances in multimedia technologies (such as compression, display, database and visualization technologies), and the increasing emphasis on multimedia applications, the production of image and video information has resulted in large volumes of images and video clips. This trend is likely to continue, and to cope effectively with the explosion of multimedia information, image-based systems have been proposed to properly organize and manage these information for rapid retrievals [3, 5, 24, 28, 30, 43].

At the same time, users of image retrieval systems are demanding for more user friendly querying facilities. There is a clear preference to access images based on their content through queries such as "Retrieve all images that look like *this*", where *this* refers to a sample image provided by the user or one that the user has painted on some sketching tool. Providing such capabilities calls for content-based image retrieval system that accesses images based on the content of the image such as color, texture and shape. In a content-based image retrieval system, the content of the database images are (semi)automatically generated and stored. A query image's content is extracted during runtime and used to match against those in the database. The result of the query is a set of images that are *similar* to the query image, rather than an exact match. Examples of content-based retrieval systems include QBIC [29], Photobook [31], Virage [1], Candid [22], VisualSEEk [39] and Chabot [30].

More importantly, content-based retrieval systems are needed in many applications. We have witnessed its use in applications such as medical imaging [22, 25], TV production [38], multimedia [40], art history [20], geology [37] and satellite image databases [23]. Other potential applications include a trademark system, an ultra-high resolution image supplier system, and an image classifier system. For the trademark system, a trademark can only be assigned if it does not resemble those in the database too closely. Likewise, a user may want to find out if his/her low resolution picture is available in the database of the image supplier system. Finally, such a system can also facilitate the classification of images [3]. For example, it has been observed that images in the *airplane* category are characterized by a large, uniform blue region located at the top portion of the image, or all across it, together with a small, dark region within it, usually centered in the image. Similarly, in a *sunset scene* category, the sky colors (e.g., purple and orange) are significant, and the existence of a distinct yellow or orange circle is also very informative.

While existing retrieval systems can handle queries about content, the technology for content-based retrieval remains an important challenge. For content-based retrieval, multiple but related features (such as color and its spatial information) can be used to select the relevant images. The importance of these features varies. In this paper, we focus on the color content. It has become clear that color plays a very important role in defining an image [12, 16, 17, 26, 29, 42, 43]. It is a local identifying feature that is independent of view and resolution, and color identification algorithm can be carried out without image segmentation and analysis. In addition, access methods indexing on color are not limited by current image analysis techniques. Moreover, color recognition can be carried out automatically without human intervention. However, it is also equally clear that color alone is not sufficient to characterize an image. For example, consider two images – one with the top halve red and bottom halve blue, while the other with the left halve red and right halve blue. Although these two images are entirely different to a human observer, they are similar in color compositions, and will be retrieved if the system is based on color compositions alone.

In this paper, we present the design and implementation of VIPER (VIsual Property-based search Engine for image Retrieval), an image retrieval system that employs both the color and spatial information of images to facilitate the retrieval process. There are two issues that need to be addressed in such a system. The first issue deals with the representation of the color-spatial information, and the similarity measure in matching images. The second issue concerns the *speedy* retrieval of relevant images since a slow system will not be useful or practical for large image databases.

For the first issue, we represent each image as a set of single-colored *clusters*. A cluster is the basic unit used in our technique and bounds a homogeneous region (in terms of color) in an image. The color of the cluster is the color of the region it bounds. In our study, each cluster is represented as a (*color, coordinates*) pair where *color* is the color of the cluster, and *coordinates* are the 4 coordinates of the rectangle representing the boundary of the cluster. To a certain extent, spatial information also captures information about shapes. Two image clusters are similar if they are of similar colors and they overlap in space. The retrieval process is based on the colors and the spatial properties of the clusters. The number of clusters used affects query efficiency as well as query correctness defined by *recall* and *precision*.[1] We study this effect and determine the most appropriate number that ensures retrieval correctness and yet maintains retrieval efficiency. Our result shows that a small number of large clusters is sufficient to find similarities between images (in terms of color-spatial information). This approach also lends itself to effectively processing queries that are based on painted images. Given a painted image with some colored objects and a large uncolored background, only the spatial distribution of the colored objects need to be used for finding matching images. Clearly, only images that have similar color in the same region will be retrieved.

For the second issue, to facilitate efficient retrieval using the color-spatial approach, we also propose a multi-tier indexing mechanism called the *Sequenced Multi-Attribute Tree* (SMAT). Each layer essentially corresponds to a feature, and can be indexed using any known indexing mechanism. The structure prioritizes the importance of the multiple features that are indexed, so that the higher layers (nearer to the root) are built on the more selective features. The search will proceed from the most selective feature to the second only if the first feature is satisfied; otherwise, the search is terminated. The process is continued with the second feature (when the first feature is satisfied) and so on. Such an index can be generalized to index any number of attributes to support retrieval that is based on a pre-defined sequence of attributes. We also addressed the issue of height-balancing in such a multi-tier indexing structure. In this paper, we implemented a two-tier SMAT where the first layer is used to prune away clusters that are of different colors, while the second layer discriminates clusters of different spatial

[1] Precision and recall are two metrics commonly used in text retrieval systems to determine the accuracy of the retrieval techniques. It has also been adopted as two of the performance metrics in image retrieval systems. Essentially, recall measures the percentage of relevant images retrieved (i.e., relevant retrieved/all relevant), while precision measures the percentage of images retrieved that are relevant (i.e., relevant retrieved/all retrieved).

locality. We conducted an experimental study on an image database of 12,000 images. Our results show the effectiveness of the proposed color-spatial approach, and the efficiency of the proposed indexing mechanism.

The rest of this paper is organized as follows. In the next section, we review some related work. Section 3 presents the proposed color-spatial retrieval technique, and Section 4 describes the proposed *sequenced multi-attribute index* that supports the color-spatial retrieval technique. In Section 5, we report on the implementation of a prototype image retrieval system and the performance study of the proposed scheme, and finally, we conclude in Section 6.

2 Related Work

Most of the existing work focused on the effectiveness of retrieval mechanisms (i.e., precision and recall). While there are some work done to address the efficiency issue (i.e., speedy retrieval), most of these work employ existing indexing mechanisms. In this section, we review related work in basic retrieval mechanisms and indexing techniques for content-based image retrieval.

2.1 Basic Mechanisms

There is a wealth of research done in developing image retrieval techniques. They can be classified into three categories: text-based retrieval, content-based retrieval, and semantic-based retrieval. Text-based retrieval techniques attempt to provide "content-based" functionalities by manual description of the image, and image access is done through keyword matching.

While such a mechanism allows abstract queries to be specified, its performance is limited by the set of keywords maintained in the system, which is usually incomplete since an image is semantically richer than text descriptions. Moreover, free-text description of an image is highly variable due to the ambiguities in the natural language associated with annotating images with text and the different interpretations of the image. In content-based retrieval technique, the content of an image (such as objects, color, texture, and shape), is used as the basis for retrieval. Such attributes can usually be automatically extracted from the images. However, these techniques are limited by the difficulty in specifying abstract queries. In the semantic-based retrieval technique, semantic meanings are used to retrieve relevant images. Typically, some form of knowledge base is required in the semantic-based retrieval systems. As such, such techniques are usually domain-specific, but can support concept-based queries [9].

Because it is able to facilitate automatic extraction of content, content-based techniques are suitable for large image databases. Existing content-based retrieval techniques can be classified into *template matching*, *global features matching*, and *local features matching*. In template matching techniques [33], templates are used to detect objects within an image. Its effectiveness can be greatly hampered by image noise, and the orientation of the search objects.

Global features such as color, texture or shape information have also been widely used to retrieve images [21, 41, 43]. Retrieval by shape information usually works well only in specialized application domains such as a criminal picture identification system where the images have very distinct shape. Color and texture are more suitable for general-purpose application domains [4, 26]. Several systems have also integrated multiple global features to improve the effectiveness of image retrieval [29, 39]. The main problem with image retrieval techniques that employ global features is that they cannot take advantage of the local features in an image. In [27], Nagasaka and Tanaka developed the color-pair matching technique to model distinct object boundaries. Chua et. al. [8] further extended the color-pair technique to perform retrieval of segmented images. Shortcomings in many of the existing color-based retrieval techniques include: the inability to recognize similar objects with different colors, the extreme sensitivity of the techniques to the scaling of an object, and the poor tolerance of noisy images.

Color-spatial information has recently received much attention. IBM's QBIC system [29] has the "positional color" feature that allows queries of the form "Find images with a red blob on the upper-right portion" to be asked. VisualSEEk [39] also combines color and position to facilitate query retrieval. Our work is more closely related to the color-spatial retrieval technique in [16]. There, two color histograms are used to generate clusters – the first represents the color compositions of the entire image, while the second represents the color compositions of the center region of an image. The important assumption is that the center of an image captures the object(s) of the image. While the technique is shown to outperform the histogram-based approach, its effectiveness is limited to images where the center region of image contains the object(s). Furthermore, because the number of clusters generated varies from image to image, and can be very large, it is not easy to develop an indexing structure to support it.

2.2 Indexing Mechanisms

Due to the large size of images and the large quantity of images, efficient and effective indexing mechanisms are necessary to facilitate speedy searching. We look at some of these schemes here.

In text-based retrieval systems, standard indexing techniques such as the signature file access methods [11], inverted file access methods [46] and direct (or sequential) file access methods have been effectively employed.

To facilitate content-based retrievals, the general approach adopted in the literature has been to transform the content information into a form that can be supported by an existing index mechanism. Shape features can be represented as geometric properties (such as shape factors, moment features and curved line features) and a geometric index (which is essentially an inverted file) can be used to facilitate fast retrievals [44]. Alternatively, shape features have also been represented as a collection of rectangles which form a rectangular cover of the shape [18]. An existing multi-dimensional indexing scheme can then be used to index the rectangular cover. Another representation of shape is the boundary information, in which case, retrieval is achieved by means of string matching and hence string matching algorithms can be used to build the index [19]. More recently, concepts from morphology are employed to map shapes of tumor in medical databases to points in a high dimensional space, and the R-tree [15] is used as the indexing structure [25].

In object-based systems, the attribute relational graph (ARG) captures the relationship between objects or parts of objects. In [32], the ARG is also mapped to points in a multi-dimensional space to exploit the use of the R-tree in indexing the images. Objects may also be represented by a signature so that signature file access methods can be employed [34].

For color, the color histogram that captures the color composition is mapped into a multi-dimensional point, and a multi-dimensional point access method (such as R-tree) is used [12]. In [26], a multi-level color histogram approach based on quad-tree structure [36] was adopted to handle positional information of color. The indexing structure used comprises 3 layers: a *dominant color classification* that splits the images into partitions based on their dominant colors, an R-tree that organizes the images within each partition based on the colors, and a *multi-level color histogram* that stores the positional information of each image.

To our knowledge, there is no indexing mechanism that has been specifically designed for handling multiple features, such as color-spatial information.

3 Color-Spatial Image Retrieval

In this section, we describe our approach to image retrieval using both the color feature and the spatial distribution of the colors. We will look at the representation of color-spatial information, their extraction, and similarity function.

3.1 Representation of Color-Spatial Information

Humans are particularly sensitive to large clusters of colors [2, 45]. Two images appear to be similar to one another if they both have regions of similar colors at roughly the same locations in the images. Thus, it makes sense to represent the color-spatial information of an image by the homogeneous regions (in terms of color) of the image, i.e., identify the set of homogeneous regions in an image and use them to facilitate the retrieval process. However, as a region may be irregular in shape, it may be cheaper to store the approximation of the region. For example, storing a rectangle that bounds a region is certainly more cost effective (in terms of storage and computation of similarity measure). As a preliminary study, we extracted single-colored *clusters* (using the proposed color-spatial technique to be discussed shortly) from three sample images, two of which are similar (as perceived by humans) and the other is not. These clusters are essentially the minimum bounding rectangles of the region they bound, and the colors assumed by the clusters are the colors of the regions they bound. Our result shows that for the two similar images, there are a few large clusters with the same color that fall in the same image space, i.e., these two images are "similar" in terms of color and spatial information. We also note that for the dissimilar images, their large clusters do not share the same color or location. Thus, we believe that a set of single-colored clusters can be effectively used to facilitate image retrieval.

3.2 Extraction of Color-Spatial Information

To extract the color and spatial information, we employ a three-phase heuristics. Our heuristics is also based on several observations. First, it is observed that the number of colors that are to be extracted need not be large. This is because most of the images are dominated by a few colors. Second, the number of clusters can also be restricted. This follows from the observation that most of the clusters of the dominating colors are large.

Our technique works as follows. In the first phase, we extract a set of *dominant* colors. A color is dominant if it occupies a large percentage of the pixels in the image. An image is represented as a 2-dimensional array of pixel values of the form $w \times h \times n$, where w and h are the number of pixels in the x- and y-axis respectively, and n is the number of bits per pixel. To compute the color content of an image, a color histogram which represents the color composition of the entire image is used. The color histogram $H(g, h)$ of image g can be obtained by counting the number of pixels with color intensity h. The color histogram is then sorted according to the number of pixels per color, and the first $C_{dominant}$ colors (most dominant) will be picked. Here, $C_{dominant}$ is some predetermined configuration set by the system (we shall delay the discussion on how it can be obtained to the end of this subsection). The reasoning behind the heuristic rule is straightforward: the color with the largest number of pixels in $H(g, h)$ is likely to be the representative color for the image, and because such a color occupies a large portion of an image space, it is likely that homogeneous regions of the color will also be large.

In the second phase, a set of clusters for each of the dominant colors are determined. The algorithm for deriving the spatial information with respect to a chosen color is based on the maximum entropy discretization method [6]. Consider the data of interest represented by n variables $(X_1, X_2, \cdots, X_n)$. The k^n-discretization process is defined as one that finds a set of k points on each axis of the n-dimensional space. These n sets of points will partition the n-dimensional space into k^n cells, R $= \{R_i | i = 1, 2 \cdots, k^n\}$, where the projection of R_i on the jth axis defines an interval $I_{i(j)}$ on X_j, $(1 \leq j \leq n)$. Let P be the probability distribution on the n-dimensional space. The process quantize P into P(R_i), $i = 1, 2 \cdots, k^n$ whose probability distribution is denoted by

$$\tau = \begin{pmatrix} R_i \\ \\ P(R_i) \end{pmatrix}, i = 1, 2, \cdots, k^n$$

The Shannon entropy H associated with τ is defined as

$$H(R) = -\sum_{1}^{k^n} P(R_i) log P(R_i)$$

The entropy discretization process is to find the sets of boundary points that partition the outcome space into cells subject to the maximization of the Shannon entropy.

We apply the maximum entropy algorithm to extract the clusters of selected colors in our images. For each selected color, the algorithm is applied to the image space in both x- and y-dimensions. The results of the application of the algorithm is a set of representative regions for each selected color. For simplicity and without loss of generality, the region R_i is represented as a rectangular space $(x^i_{min}, y^i_{min}, x^i_{max}, y^i_{max})$. After discretization, an event covering process is used to detect relevant spatial information. The algorithm works as follows. Initially, the entire image is regarded as one whole region. During the first pass, the image is partitioned into four regions based on the maximum entropy discretization criterion. For each region, an evaluation criterion is used to determine whether further partitioning is needed. If the observed samples that fall within the region deviate significantly than the expected frequency, the region needs to undergo further partitioning. The expected frequency can be computed using the uniformity assumption method: If no outcome is expected to occur more than the others, the probability distribution is uniform. The expected frequency of observations in a cell is estimated by

$$M \times \frac{\mu(I_{x_i})}{\mu(I_x)} \times \frac{\mu(I_{y_i})}{\mu(I_y)}$$

where M is the total number of data; I_{x_i} and I_{y_i} are the projected intervals of the cell i on the x- and y-axis respectively; and I_x and I_y are the extended intervals that enclose I_{x_i} and I_{y_i}, and $\mu(I_{x_i})$ is the average or the expected value of I_{x_i}.

Once the expected frequencies are calculated, the deviation can be computed by

$$D = \frac{(obs(i) - exp(i))}{\sqrt{(exp(i))}}$$

If D exceeds a certain threshold, then the cell is covered and no further partitioning is needed. Otherwise, it is placed in a stack for subsequent passes. The partitioning process stops when either all the cells are covered or the number of samples within a cell falls below the given limit.

At the end of phase two, we have a large set of single-colored clusters. In phase three, we rank the clusters (regardless of color) in descending order of their sizes (i.e., area of the clusters). The $Cl_{dominant}$ largest (dominant) clusters will be picked to represent the color-spatial information of the image.

From the above discussion, we note that the number of clusters can affect performance. A small number of clusters can speed up the retrieval process and lower the storage cost at the expense of the retrieval accuracy. Adding more clusters may increase the retrieval accuracy since these clusters are likely to lead to more retrievals. However, having too large a number of clusters will eventually lead to poor performance. This is because the set will contain small clusters which may contribute to the retrieval of irrelevant images. Further, the storage of these large number of spatial rectangles, and retrieval and comparison cost are likely to become high too. Therefore, it is important to fine-tune $Cl_{dominant}$ so that a compromise between retrieval effectiveness and retrieval efficiency can be obtained. To investigate the effect of $Cl_{dominant}$ on retrieval accuracy, we conducted various experiments using a large collection of about 12,000 images of various categories and classifications. We varied the number of dominant regions, $Cl_{dominant}$, from 3 to 8 and compute their corresponding *normalized* precision and recall. We will look at the experiments in Section 5. It suffices for us to point out that our analysis shows that the retrieval effectiveness (in terms of precision and recall) is good enough with only 4 clusters.

Now, let us come back to the issue on the choice of an appropriate value for $C_{dominant}$ in phase one. Since we are only interested in $Cl_{dominant}$ clusters, we need to pick at most $Cl_{dominant}$ colors in the first phase, i.e., $C_{dominant} \leq Cl_{dominant}$. For simplicity, we set $C_{dominant} = Cl_{dominant}$.

3.3 Similarity Function

Now, let us look at the similarity function used for image retrieval, which is similar to the one used in [16]. Given the color-spatial information, let $C(i,k)$ denote the ith cluster of color k. The similarity measurement (to match objects of the same color), L, between two images g_1 and g_2 is defined as follows:

$$L_{g_1,g_2} = \sum_{k=1}^{N_c} \sum_{i=1}^{N_k^{g1}} \sum_{j=1}^{N_k^{g2}} C_{g_1}(i,k) \cap C_{g_2}(j,k)$$

where N_c is the total number of colors in the representative set, $N_k^{g_i}$ is the total number of clusters of color k in image g_i $(i = 1,\ 2)$, and $C_{g_1}(i,k) \cap C_{g_2}(j,k)$ is the intersection between cluster $C_{g_1}(i,k)$ and cluster $C_{g_2}(j,k)$.

To compute the intersection between two clusters, we determine whether the two clusters overlap. Let the two clusters be $C_1 = ((x_1, y_1)(x_2, y_2))$ and $C_2 = ((x_3, y_3)(x_4, y_4))$. If the two clusters do not overlap, $C_1 \cap C_2 = 0$. Suppose the two clusters do overlap, the degree of overlap is computed by first sorting the x coordinates and the y coordinates of the two clusters. Let $\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \tilde{x}_4$ and $\tilde{y}_1, \tilde{y}_2, \tilde{y}_3, \tilde{y}_4$ denote the sorted lists of x and y coordinates respectively. The intersection region can be computed as follows:

$$C_1 \cap C_2 = (\tilde{x}_3 - \tilde{x}_2) * (\tilde{y}_3 - \tilde{y}_2)$$

3.4 Image Retrieval Process

The image database is initially preprocessed to determine the clusters (color-spatial information) of the images. Given a sample query image, *at most* Cl_{query} clusters are extracted, where Cl_{query} is a predetermined value. Note that it is possible to have fewer than Cl_{query} clusters because a painted query may contain just a single-colored object such as an orange sun. The color-spatial information of each image in the database is then compared with those of the query image using the similarity function described above. The images can then be ranked based on the percentage of overlap, retrieved and displayed in that order.

4 Sequenced Multi-Attribute Tree: A Color-Spatial Index

Having described the color-spatial technique, there remains the problem of speedy image retrieval. With tens of thousand of images, it is not practical to sequentially test each image in the database against the sample image. Even though our approach restricts the number of clusters to $Cl_{dominant}$, the number of cluster comparisons to be performed is still very large, about $N \cdot Cl_{dominant} \cdot Cl_{query}$ where N is the number of images in the database, and $Cl_{dominant}$ and Cl_{query} are the number of dominant clusters selected for the database images and the query image respectively. Since only a small number of images is likely to match the sample image, a large number of unnecessary comparisons are being performed. As the comparison is expensive, we propose an index structure, the *Sequenced Multi-Attribute Tree* (SMAT), to provide an efficient means to reduce the number of comparisons. SMAT is based on three observations on the similarity function of the color-spatial approach:

1. Color must be matched before the spatial property as color is deemed a more important feature.

2. If two clusters of two images share the same spatial property but different color content, then the two clusters will not contribute to the similarity function.

3. If two clusters of two images share the same color but with non-overlapping spatial properties, then the two clusters will also not contribute to the similarity function.

4.1 The Sequenced Multi-Attribute Tree

The *sequenced multi-attribute tree* (SMAT) is a multi-tier tree structure, where each layer corresponds to an indexing attribute. For example, the top layer can be based on color, the second is based on color percentage or size of the cluster, and the last is based on spatial property. Each layer can be constructed using any indexing mechanism. For example, the top layer can be implemented using a single dimensional indexing structure such as the B^+ tree [10]. On the other hand, the lowest layer can employ a multi-dimensional indexing structure like the R-tree [15]. Except for the lowest level, entries in the leaf nodes of all levels point to the roots of the trees in the next level. Only the leaf nodes of the lowest level tree contain pointers to the image data. Thus, SMAT essentially consists of multiple trees integrated together in a hierarchical manner. Such a structure can facilitate speedy retrieval of images. To reach to the lowest layer of SMAT where the actual images are pointed to, the query must satisfy the conditions relating to the discriminating keys in all the higher layers. Any condition violated in any layer will terminate the search path prematurely. Thus, SMAT can reduce the search space by pruning off redundant paths early. Multi-tier indexes have been implemented for other applications. For example, in temporal databases, there is a two-level Nested ST-tree [14] where the top level is a B^+-tree while the second level is an AP-tree. However, in these applications, queries can target at any attribute in any order. In our application, the sequencing of the attributes in SMAT depends on the selectivities of the attributes, and a careful choice of sequence is critical to performance. For example, searching based on color and spatial information is dominated by color, and hence pruning by color first would be better. Such search is also common in applications such as medical systems and data mining.

In this study, as a first cut, we have implemented a two-tier SMAT structure. Figure 1 shows the structural view of the SMAT structure implemented. We employ a variation of the R-tree structure to implement our SMAT. The first layer discriminates clusters based on color. Since color is a single-dimensional attribute, the R-tree used at this layer is a single-dimensional R-tree (1-D R-tree). Each entry has a color range that defines the data space of the subtree pointed by its child pointer. The color ranges of internal nodes do not overlap, unless they are exactly the same range. This occurs only when the data is very skewed. Entries of the leaf nodes of the first layer R-tree are of the form (*color-range, BR, PTR*), where *BR* defines the spatial bounding rectangle which contains all the clusters' color rectangles within the image space, and *PTR* points to an R-tree of the next layer. Spatial information is necessary in the leaf nodes of this layer for balancing purposes. Suppose for a given color range, the next layer R-tree pointed by *PTR* outgrows others and the next split involves its root node (*PTR*). By splitting such a node, we will continue increasing its height. To enable some form of balancing, we split the node according to the splitting strategy adopted at the second layer, but insert the entries into the leaf node of the first layer instead. In other words, we create two entries of the same color range (at the first layer), but with different bounding rectangles. We shall discuss the load-balancing issues shortly.

The second layer is based on the spatial information of the clusters. Each entry of the internal node contains a rectangle that defines its child node's data space and a pointer pointing to the subtree. The second layer R-tree is like a 2-dimensional spatial R-tree (2-D R-tree) structure. For the leaf nodes, entries are of the form (*color, coordinates, PTR*). The *color* attribute contains the color of the cluster, the *coordinates* attribute contains the four

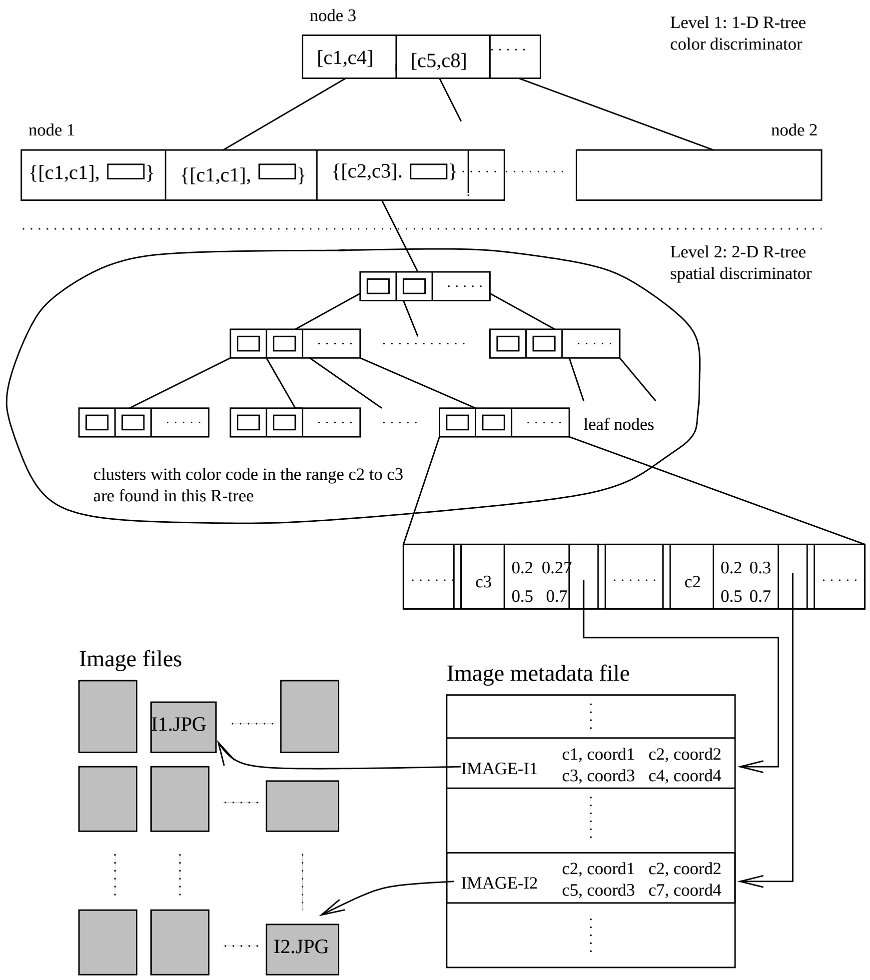


Figure 1: The SMAT structure.

coordinates of the cluster, and PTR is a pointer to the address in the database (of file) that contains the metadata of the image (see Figure 1). The file contains the ID of the image, and the colors and coordinates of the $Cl_{dominant}$ dominant clusters. These information are used in computing the similarity function (we shall see how these are used when we discuss the matching algorithm).

4.2 Matching and Searching an SMAT

Figure 2 shows the algorithm for retrieving the images that are similar to a sample image. Given a sample image, the algorithm extracts Cl_{query} dominant clusters (line 1). For each of the clusters extracted, it determines the set of images that are similar to it (lines 2-12). This is done by traversing SMAT to determine the clusters that match those of the sample image. It suffices to know that the *search* algorithm returns a list of pointers to the metadata file that contains information on potential matching images. Recall that these information include the image ID and the (*color, cluster*) pairs of the image. From these information, the algorithm proceeds to compute the similarity value of the sample image and the candidate image (line 7). Note that the function *computeSimiarity* computes the similarity for two images based on their clusters since it has all the necessary information (Cl_{query} clusters of the query image, and $Cl_{dominant}$ clusters of the candidate image obtained from the metadata file). Thus, it is able to rank the candidate image accordingly (line 8). Since it is possible that other clusters of the sample image may also match the same candidate image at a later iteration, the image IDs are maintained in a hash table to avoid subsequent comparisons and retrieval (line 5,10). Finally, all the images can be retrieved based on the image IDs (line 13).

The search algorithm of an SMAT structure is fairly straightforward, and follows from the way an R-tree is searched. The algorithm descends the 1-D R-tree from the root, and at each internal node, entries are checked. For each color range that contains the search color, the subtree is searched. When a leaf node is reached, the color of the search cluster is used to check for entries whose color ranges contain the color. For all color ranges that qualify, their spatial bounding rectangles are checked to see if they intersect the search cluster. For qualified

Algorithm Match
Input: Sample image, IMG
output: Images similar to IMG

```
cluster ← extractCluster(IMG, Cl_query)
for each cluster, cl ∈ cluster {
    PTR-list ← search(cl, SMAT.root)
    for each ptr in PTR-list {
        if ptr has not been examined {
            dbImageCluster ← retrieveInfo(ptr)
            similarity ← computeSimilarity(dbImageCluster, cluster)
            rank(image_id, similarity)
            cache(ptr)
        }
    }
}
retrieve the images using the image IDs.
```

Figure 2: Algorithm Match.

entries, the search continues to the corresponding 2-D R-trees at the next layer. While the traversal of the 1-D R-tree often leads to a distinct path (unless there are duplicates), more than one subtree under the 2-D R-tree may need to be searched. Nevertheless, the search algorithm can eliminate irrelevant clusters of the indexed images and examine only clusters near the search area.

4.3 Inserting Image Clusters into SMAT

Inserting image clusters into a SMAT structure raises some interesting issues concerning the growth of the tree. The first issue concerns the *initial loading* of SMAT. In this case, the tree is not "mature" in the sense that not all layers may have been constructed. The question of when SMAT grows from one layer to the next arises. The second issue deals with the height-balancing of SMAT. While R-tree is height-balanced, SMAT may not be fully height-balanced as images may be inserted towards one end of SMAT. We proposed that SMAT grows downward until some criterion is met, and it grows upward when height-imbalance occurs. We shall discuss this solution in this section.

Initially, we predetermined the height of all the layers. For a SMAT structure with k layers, $L_1, L_2, \cdots, L_k$, let the predetermined height for layer L_i be h_i. Note that the height will change dynamically as SMAT grows. During initial loading, SMAT is not fully developed, and so the height is used to guide its growth downward as follows: layer L_{i+1} will appear only if all the nodes along the path leading to the leaf node of layer L_i in which the new record is to be inserted are full, and the length of the path has reached the threshold height, h_i. This is to ensure that the height of SMAT is maintained and not increased further unless necessary. To illustrate, consider the 1-D R-tree in Figure 1. Suppose leaf node 1 is full, and h_1 is set to 2, and a new cluster is to be inserted into leaf node 1. If node 3 is full, instead of allowing the 1-D R-tree to grow, SMAT grows downward by creating the next layer tree, and the record is inserted there. On the other hand, if node 3 is not full, then creating the next layer will undoubtedly increase the height of the search path by one. Instead, leaf node 1 should be split as normal.

Once all the layers of SMAT are developed, the issue of height-balancing becomes a concern since it affects the retrieval time of SMAT. Although the R-tree is height-balanced, SMAT may not be so. For example, if most records are added to the leftmost path of the tree, this may result in a tree being skewed to the left. This happens especially if there are a lot of clusters of a particular color. Thus, there is no guarantee that all the trees in the second layer index will grow and shrink at the same rate. This means that it is possible that a particular tree in a level may grow much faster than the other trees in the same level causing SMAT to be skewed to one side. This is to say that the basic SMAT structure can only be locally but not globally height-balanced.

Since SMAT is a multi-tier structure, the concept of height-balanced is slightly different from a single structure index. A SMAT structure is height-balanced if the following two conditions are met:

1. Each tree structure within a layer is height-balanced.

2. The difference in the heights of trees within a layer, say L_i, is at most e_i for some predetermined e_i for each layer.

Figure 3 illustrates a height-balanced tree. As can be seen, in the worst case, the difference between trees in height within a k-layer SMAT is $\sum_{i=2}^{k} e_i$ (assuming that layer 1 employs a height-balanced index structure).

To keep SMAT height-balanced, we allow the upper layers to grow once the lowest layer has been established. We keep track of the minimum height of the trees at each layer. If there is an increase in the height of a tree (at a layer) as a result of an insertion, the new height of the tree is compared against the minimum height at that layer. If the difference between the two is above a certain predetermined threshold, then rebalancing is activated. Rebalancing is performed as follows. Let the layer where rebalancing is needed be L_i, and its parent layer be L_{i-1}. Let the root of the tree that causes height imbalance at L_i be R_i, and the leaf node of L_{i-1} that points to R_i be LN_i. Let the entry in LN_i that points to R_i be l_{old}. There are two

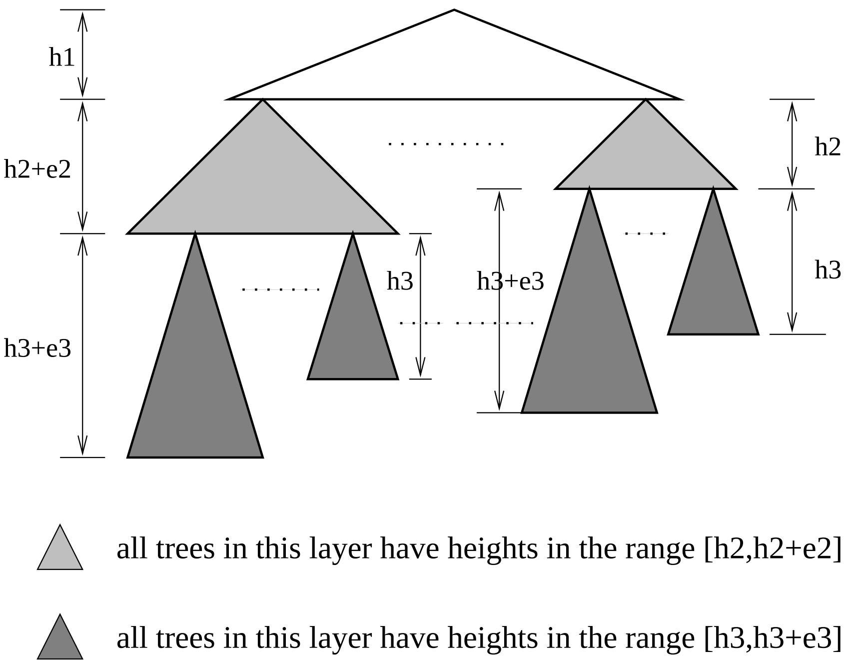


Figure 3: A height-balanced SMAT.

cases to consider:

- If the color component of l_{old} is a range of colors, say c_1—c_k, then we create a new entry in LN_i, l_{new}, and split the color range of l_{old} into two equal halves: c_1—c_i for l_{old} and c_{i+1}—c_k for l_{new}. Records rooted at R_i are then reinserted to layer L_i according to the respective ranges. Note that this process may cause layer L_i to violate the height-balanced property as one of the trees may be shorter than the existing shortest tree. The effect of height-balancing is then propagated to other trees in the layer.

- If the color component of l_{old} is a single value, i.e., c_1—c_1, then the information at R_i is used to insert a new entry, l_{new} into LN_i. l_{old} is set to point to the left child of R_i, and l_{new} is set to point to the right child of R_i. R_i can then be removed. Note that the corresponding bounding information in l_{old} needs to be updated too.

As a first cut, we have only implemented the second technique. In this case, when the color component of l_{old} is a range, the range is replicated across l_{old} and l_{new}.

The insertion algorithm which SMAT adopts within a tree is similar to that used in R-tree in that new clusters are added to the leaves, nodes that overflow are split, and splits are propagated up the tree. The splitting algorithms adopted are based on the quadratic-cost algorithm of R-tree by Guttman [15] for the second layer, and that of B^+-tree for the first layer. The algorithm attempts to find a small-area split, but is not guaranteed to find one with the smallest area possible. There is, however, the additional task of handling height-balancing.

In our study, we use only 256 colors, and so it suffices for us to restrict the threshold height during initial loading to 2 levels. For the 2-D R-tree, we also restrict the difference in height within the layer to be the height of the shortest tree in the layer.

5 System Design, Implementation and Evaluation

In this section, we describe the system design and implementation of VIPER (VIsual Property-based search Engine for image Retrieval), a prototype image retrieval system that is based on the proposed techniques. We also present two sets of experiments we have conducted to evaluate the system. The first experiment studies the retrieval effectiveness (accuracy) of the proposed color-spatial technique, while the second is on the retrieval efficiency of the proposed SMAT structure.

5.1 System Design

VIPER is designed and implemented on the UNIX system using C programming language. The overall system architecture is shown in Figure 4. It consists of two main modules: the *preprocessing* and *retrieval* modules. Initially, the image database is preprocessed by the *preprocessing* module to extract the color-spatial information. The *retrieval module* accepts sample images from the user as queries through a *graphical user interface* module, extracts the color-spatial information from the sample image, and traverses the SMAT structure for matching images. Besides the SMAT structure, we also used an R-tree structure for purpose of comparison.

The image database holds the set of 12,000 images used for testing. The images, together with the associated text descriptions, are acquired commercially from Kagema Corporation. They have been divided into eighteen categories by the image supplier. Major categories include: animal, art, computer, food, nature, travel, etc. For purpose of this study, we have normalized the images to thumbnail size (either 115×174 or 174×115 pixels). These images are preprocessed to extract the appropriate color-spatial information, and are stored in the SMAT structure.

The graphical interface is designed to display the top 60 retrieved images ranked in decreasing order of similarity. Queries are issued by image example. Users are able to specify content-based queries by selecting a sample image or a painted image. Figure 5 shows the user interface implemented, and a sample query with some of its results. The interface also allows the user to "click" on an image to find out more detailed information about the image (as shown in the figure). Note that in our implementation, we did not display the query image

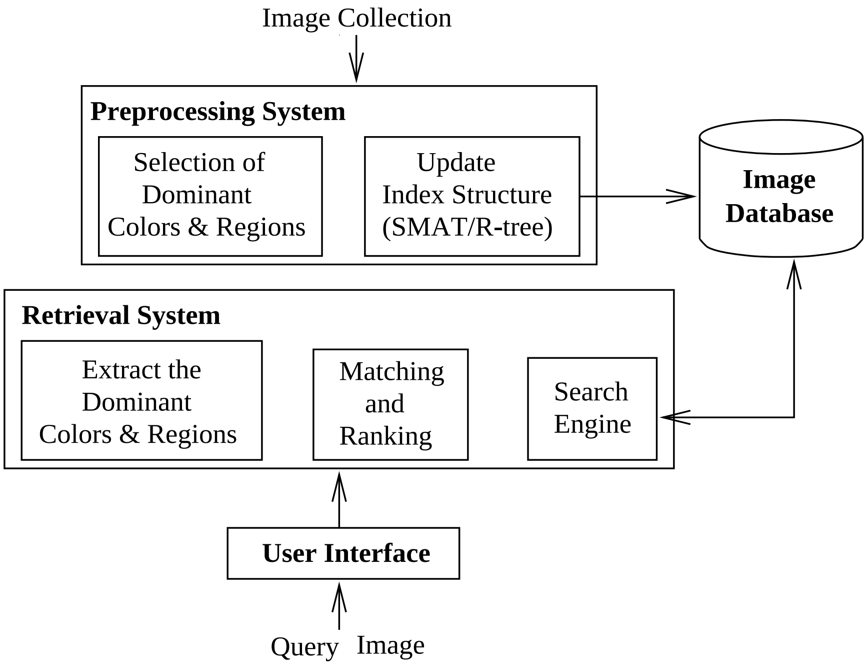
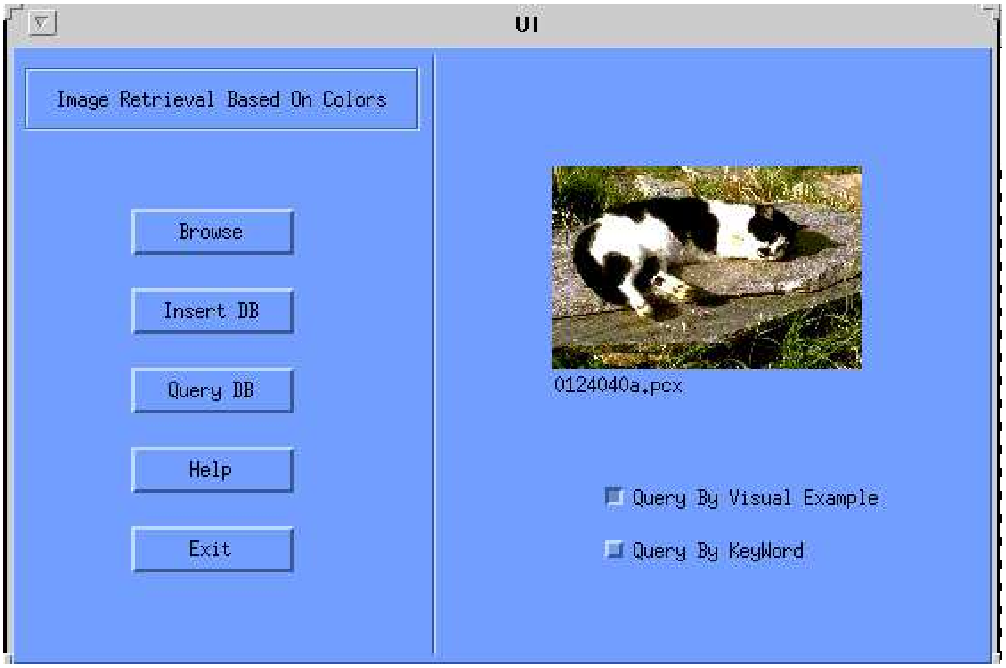


Figure 4: The architecture of VIPER.

since it will always satisfy the query. As can be seen from this sample query image, the system is able to retrieve relevant images (in terms of color-spatial information). It should also be pointed out that we have chosen the sample image because of its more complex color compositions, rather than using a less complex one for illustration. Retrieval can also be performed through a sketching tool that we have developed. Figure 6 shows the interface for painting, a sample query (obtained by loading an existing image, and erased the background away) and some of its results.

Besides the color-spatial technique, VIPER also supports structured retrieval, that is retrieval based on fixed attributes like title, date and keywords. These are the attributes commonly used for retrieval. This will enable queries like "Retrieve all pictures taken before 1944" to be supported. To speed up the access of images using these fixed attributes, indexing methods are used. The index that we used for the structured attribute such as date and title is the



(a) Query screen.

(b) Sample result screen.

Figure 5: Sample query and resultant images.

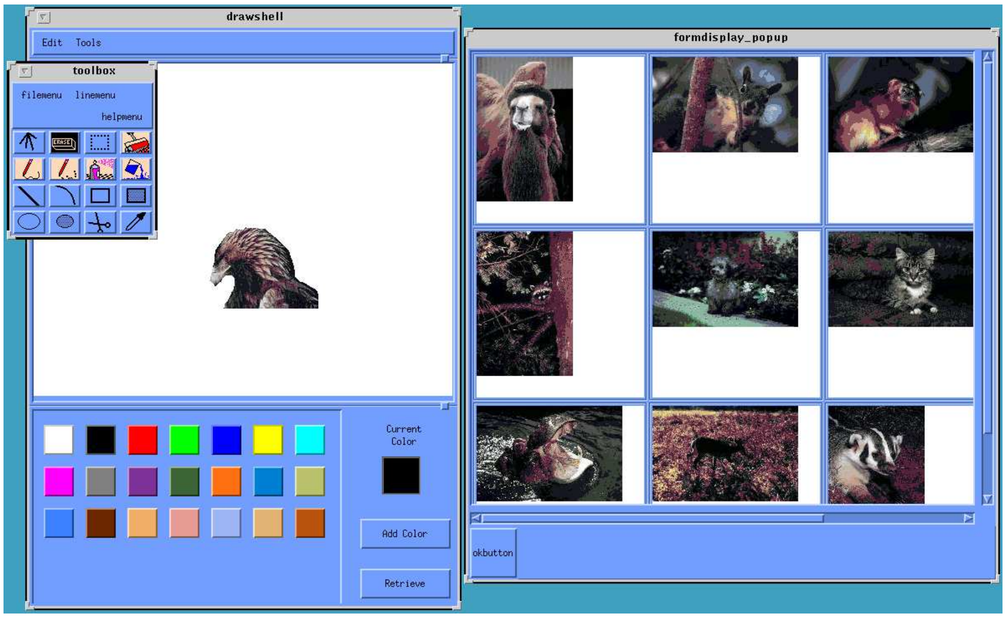


Figure 6: Retrieval through sketching tool.

(a) Class A image.

(b) Class B image.

(c) Class C image.

(d) Class D image.

(e) Class E image.

Figure 7: Sample images from the various classes.

B^+-tree. For completeness, free-text based retrieval is also incorporated. We want the user to be able to retrieve images based on descriptions such as "Retrieve all pictures with a swimsuit model". The inclusion of such retrieval mechanism greatly improves the functionality of the system. As signature file access method has been shown to be efficient, we use signature file access method to organize the free-text.

In our experimental study, we have *turned off* the text-based mechanisms in order for us to study the proposed color-spatial techniques in isolation.

5.2 Test Queries

As mentioned, our image database contains about 12,000 images and covers a wide range of categories. To study the effect of certain types of images on our system performance, we classify the images into five categories, and representative images from these five categories are picked as sample queries. The five categories are as follows:

- Class A. Images which belong to this class basically do not have any distinguishable center object. Examples of images that belong to this class can be scenery, paintings or images of repeated patterns or uniform color. One distinguishable observed characteristic of this class of images is that the dominant clusters that are selected are of the same color. An example is shown in Figure 7(a).

- Class B. Images which belong to this class basically have a small distinguishable center object. A center window of size 12% that of the image space is used. By small, we mean that more than 60% of the center object is within the center window. Figure 7(b) shows an example.

- Class C. Images which belong to this class basically have a medium distinguishable center object. In this case, the center window is 50% of the image space, and an object is of medium size if more than 60% of it is within the center window. An example of an image that belongs to this class is shown in Figure 7(c). For this class of images, as the center object is large enough, it is likely that the third or fourth largest cluster of the Cl_{query} dominant clusters chosen is of colors belonging to the center object. Thus we expect the precision to be better in this case since the object color is taken into consideration.

- Class D. This class of images consists of large distinguishable center object. That is to say, more than 60% of the center object are within a center window that is 85% of the image space. A good example of this class of images is shown in Figure 7(d). For this class of images, the second largest cluster of the Cl_{query} dominant clusters chosen is already of colors belonging to the center object.

- Class E. This class of images will be those images that do not belong to the above four classes mainly because the object found is not within a center window but instead is located in one of the four quadrants of the images (see Figure 7(e)).

A total of ten images, two from each class, are picked as query images. Because our database is relatively large, the number of relevant images for each query is obtained using keyword search, followed by visual inspection of the images retrieved. Each image in the database has a list of keywords supplied by the supplier of the database and this list is found to be precise in describing the contents of the images. For simplicity, we also set Cl_{query} to $Cl_{dominant}$.

5.3 Normalized Recall and Precision

It is not useful to retrieve all relevant images in an image collection of over 12,000 images. In our experiment, we evaluate only the top 60 images ranked in terms of the similarity measure. Since not all relevant images are retrieved, the commonly used evaluation measures such as recall-precision tables or graphs are not appropriate. Instead, we compute the normalized precision (P_n) and normalized recall (R_n) [35] for each query. The formulas for normalized precision (P_n) and normalized recall (R_n) are given below, where N is the number of images in the collection, R is the number of relevant images in the collection and $rank_i$ is the rank order of the ith relevant image s retrieved by the system.

$$P_n = 1 - \frac{\sum_{i=1}^{R}(log\ rank_i - log\ i)}{log\ \frac{N!}{(N-R)!R!}}$$

$$R_n = 1 - \frac{\sum_{i=1}^{R}(rank_i - i)}{(N-R)R}$$

In our computation, we set N to 1200 by assuming that all relevant images are ranked within the top 1,200. This is reasonable as most relevant images were found to be ranked among the top 200. For the relevant images that are not ranked among the top 60, their ranks are set to near the mid way between 600 and 1200. Assuming the mid way point between 600 and 1,200 is avg_rank, relevant images that are not ranked among the top 60 are ranked in our computation as avg_rank, avg_rank+1, avg_rank+2 and so on.

5.4 On Selection of $Cl_{dominant}$ Dominant Clusters

This experiment investigates the "optimal" value of $Cl_{dominant}$ that will give a satisfactory performance under each class of images. We compare our approach against the integrated approach in [16]. The approach (see Section 2.1 for a brief description) maintains and keeps track of all clusters that are extracted, and has been shown to be better in terms of both recall and precision over the color histogram approach. Besides determining the normalized precision and recall metrics, we also computed the change in normalized precision and recall for each query, which is calculated using the following formulas:

$$\Delta(P_n) = \frac{P_n(Cl_{dominant}) - P_n(Integrated)}{P_n(Integrated)} \times 100\%$$

$$\Delta(R_n) = \frac{R_n(Cl_{dominant}) - R_n(Integrated)}{R_n(Integrated)} \times 100\%$$

The normalized precision and recall for these experiments are summarized in Tables 1 and 2. The change in normalized precision and recall for each query are summarized in Table 3.

From Tables 1 and 2, it is obvious that $Cl_{dominant} = 4$ results in significant improvements in the normalized precision and recall for many of the queries tested. The normalized precision and recall for $Cl_{dominant} = 4$ averaged over the set of queries are 0.345 and 0.642. Surprisingly, large number of representative spatial rectangles do not produce higher recall/precision. When the number of rectangles is large, we have additional smaller rectangles, and these smaller rectangles lead to the retrieval of less relevant images, i.e., images that do not match on the large clusters but on the small clusters. The same is true for the integrated approach. In fact, the integrated approach is worse because half the number of clusters are based on the object (which is assumed to be at the center of the image). The improvement in retrieval

		$Cl_{dominant}$						
Class	Query	3	4	5	6	7	8	integrated approach
A	Q1	0.380	0.459	0.399	0.333	0.328	0.338	0.368
A	Q2	0.330	0.310	0.285	0.346	0.304	0.238	0.182
B	Q3	0.429	0.477	0.410	0.406	0.362	0.302	0.268
B	Q4	0.265	0.393	0.348	0.370	0.357	0.340	0.615
C	Q5	0.154	0.145	0.147	0.148	0.127	0.146	0.105
C	Q6	0.249	0.216	0.220	0.222	0.222	0.206	0.174
D	Q7	0.452	0.274	0.223	0.190	0.157	0.187	0.127
D	Q8	0.382	0.372	0.370	0.368	0.374	0.373	0.389
E	Q9	0.422	0.435	0.411	0.410	0.375	0.387	0.259
E	Q10	0.311	0.368	0.374	0.388	0.390	0.308	0.227
Avg		0.338	0.345	0.319	0.319	0.300	0.283	0.271

Table 1: Normalized precision for different $Cl_{dominant}$ values.

performance for $Cl_{dominant} = 4$ are 44% for normalized precision and 10% for normalized recall (see Table 3). Since $Cl_{dominant} = 4$ performs the best on average, we shall use this value throughout our experiments.

5.5 On the Sequence Multi-Attributes Tree

The purpose of this experiment is to compare and contrast the retrieval efficiency of SMAT with the standard R-tree and at the same time, verify the practicality of SMAT in the indexing of an image database. For the R-tree method, each color-cluster is represented by (color-id, x_{min}, y_{min}, x_{max}, y_{max}), i.e., a 2-d rectangle in 3-d space. In this study, two page sizes are tested — 1K and 2K. These page sizes are chosen so that we have index trees of reasonable height. The maximum number of entries per page is shown in Table 4.

For each image, we use 4 rectangles, and so we have a total of 48,000 rectangles. For each class of images, we picked 10 queries as query images. The number of pages touched is

Class	Query	$Cl_{dominant}$ 3	4	5	6	7	8	integrated approach
A	Q1	0.670	0.721	0.695	0.645	0.645	0.645	0.645
A	Q2	0.620	0.620	0.601	0.620	0.602	0.565	0.545
B	Q3	0.696	0.746	0.695	0.695	0.670	0.620	0.594
B	Q4	0.581	0.657	0.619	0.632	0.632	0.619	0.784
C	Q5	0.518	0.518	0.518	0.518	0.505	0.517	0.492
C	Q6	0.573	0.553	0.553	0.553	0.553	0.543	0.522
D	Q7	0.716	0.594	0.574	0.554	0.533	0.554	0.513
D	Q8	0.662	0.662	0.662	0.662	0.662	0.662	0.663
E	Q9	0.681	0.695	0.681	0.681	0.652	0.666	0.594
E	Q10	0.605	0.650	0.650	0.650	0.650	0.605	0.560
Avg		0.632	0.642	0.625	0.621	0.611	0.600	0.591

Table 2: Normalized recall for different $Cl_{dominant}$ values.

recorded, and the average number of pages touched in each class is used for comparison. The relative performance between SMAT and R-tree for the two page sizes is shown in Figure 8. In the figure, a value of 1 indicates that SMAT is as good as the R-tree. A value greater than 1 implies that SMAT is worse than the R-tree, and a value less than 1 means that SMAT outperforms the R-tree.

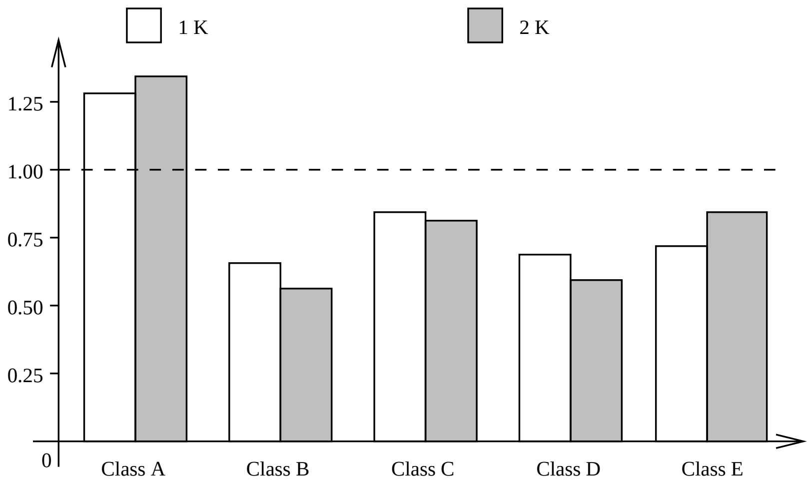


Figure 8: Relative performance of page accesses

From the above results, it is clear that SMAT performs better in terms of the total number

		$\Delta(P_n)$						$\Delta(R_n)$					
Class	Query	3	4	5	6	7	8	3	4	5	6	7	8
A	Q1	3	25	8	-10	-11	-8	4	12	8	0	0	0
A	Q2	81	70	57	90	67	31	14	14	10	14	10	4
B	Q3	60	78	53	51	35	13	17	26	17	17	13	4
B	Q4	-57	-36	-43	-40	-42	-45	-26	-16	-21	-19	-19	-21
C	Q5	47	38	40	41	21	39	5	5	5	5	3	5
C	Q6	43	24	26	28	28	18	10	6	6	6	6	4
D	Q7	256	116	76	50	24	47	40	16	12	8	4	8
D	Q8	-2	-4	-5	-5	-4	-4	0	0	0	0	0	0
E	Q9	63	68	59	58	45	49	15	17	15	15	10	12
E	Q10	37	62	65	71	72	36	8	16	16	16	16	8
Avg		53	44	34	33	24	18	9	10	7	6	4	2

Table 3: Change in normalized precision and recall for different $Cl_{dominant}$ values (in %).

Page size	Max. leaf entries/page	Max. non-leaf entries/page
1K	12	36
2K	24	72

Table 4: Characteristics of the SMAT structure.

of pages accessed all the time except when class A images are used as the query image. Two reasons account for this. First, as pointed out earlier, the dominant clusters of a Class A image are of the same color. As such, more clusters are skewed towards the colors. Second, in SMAT, the height-balancing technique adopted is the less sophisticated one that replicates the color range rather than splitting the color range. This leads to more trees being searched. We believe by incorporating the other mechanism can improve the performance of SMAT over the R-tree for Class A images. SMAT is particularly good for Class B and Class D images, with only 60% of the page accesses of the R-tree. For Class C and Class E, its page accesses is reduced to 85% that of the R-tree. Such improvements might be better if more attributes are brought into the index structure and if a larger image database is used.

6 Conclusion

In this paper, we have presented a color-spatial technique for image retrieval. We have also proposed a new indexing structure, called the sequenced multi-attribute tree (SMAT), to support the color-spatial technique. The index can be generalized and adopted for other applications where attributes indexed can be ordered based on their selectivities or significance. We have implemented a prototype system that employs the proposed techniques. Our experimental results on an image database of 12,000 images demonstrated the effectiveness of the color-spatial technique in terms of normalized recall and precision, and the efficiency of the SMAT structure in terms of number of pages touched.

There are several directions in which we plan to continue in this work. First, we would conduct more experiments on our data sets as well as other image databases to further substantiate the results of this work. We want to see if our result that a small number of clusters is sufficient holds for other image databases. Other experiments will include the following: more queries, larger collection of images, incorporate a better height-balancing algorithm, incorporate perceptually similar colors, and comparative study with other techniques. Second, we would explore the use of color as well as texture to determine the clusters such that it can be used to capture more information about the shape of objects in an image. One simple example is a tree with green leaves and brown branches appearing in between. Due to the branches or yellowish leaves, instead of one huge cluster which our human eyes are likely to identify with, we get many small clusters that describe slightly different color. With the use of texture and some deductive rules, a big cluster may be obtained. Such representation are important for systems that intend to support queries by sketch. Third, we would examine the effectiveness of incorporating other features (such as discriminating by percentage of colors, area of clusters) into the SMAT structure. Finally, we will compare our technique with recently proposed color-spatial techniques [7], and other indexing structures such as FastMap [13].

Acknowledgement

Chin Chin Goh and Wan-Meng Lee implemented the SMAT, and part of VIPER. This research is partially supported by the university research grant RP950694.

References

[1] J. R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R.C. Jain, and C. Shu. The virage image search enginee: An open framework for image management. In *SPIE Proceedings of the Storage and Retrieval for Still Images and Video Databases IV*, pages 76–86, February 1996.

[2] J. Beck. Perceptual grouping produced by line figures. *Percept. Pyschophys.*, 2:491–495, 1967.

[3] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Recognition of images in large databases using color and texture. In *http://elib.cs.berkeley.edu/papers.html*, 1997.

[4] E. Binaghi, I. Gaglardi, and R. Schettini. Indexing and fuzzy logic-based retrieval of color images. In *Visual Database Systems, II, IFIP*, pages 79–92, 1992.

[5] S.K. Chang and A. Hsu. Image information systems: Where do we go from here? *IEEE Transactions on Knowledge and Data Engineering*, 4(5):431–442, October 1992.

[6] D.K.Y. Chiu and T. Kolodziejczak. Synthesizing knowledge: A cluster analysis approach using event-covering. *IEEE Transactions on Systems, Mana nd Cybernetics*, 16(2):462–467, 1986.

[7] T. S. Chua, K. L. Tan, and B. C. Ooi. Fast signature-based color-spatial image retrieval. In *Proc. of the International Conference on Mutlimedia Computing and Systems'97.* June 1997.

[8] T.S. Chua, S.K. Lim, and H.K. Pung. Content-based retrieval of segmented images. In *Proceedings of the 1994 ACM Multimedia Conference*, pages 211–218, October 1994.

[9] T.S. Chua, K.C. Teo, B.C. Ooi, and K.L. Tan. Using domain knowledge in querying image databases. In *Proceedings of the 3rd International Conference on MultiMedia Modeling*, pages 497–498, Toulouse, France, November 1996.

[10] D. Comer. The ubiquitous b-tree. *ACM Computing Surveys*, 11(2):121–137, June 1979.

[11] C. Faloutsos. Signature files: Design and performance comparison of some signature extraction methods. In *Proceedings of the ACM SIGMOD Conference*, pages 63–82, Austin, Texas, May 1985.

[12] C. Faloutsos, W. Equitz, M. Flickner, W. Niblack, D. Petkovic, and R. Barber. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3):231–262, 1994.

[13] C. Faloutsos and K-I Lin. Fastmap: A fast algorithm for indexing, data mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD Conference*, pages 163–174, May 1995.

[14] H. Gunadhi and A. Segev. Efficient indexing methods for temporal relation. *IEEE Transactions on Knowledge and Data Engineering*, 5(3):496–509, June 1993.

[15] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD Conference*, pages 47–57, May 1984.

[16] W. Hsu, T.S. Chua, and H.K. Pung. An integrated color-spatial approach to content-based image retrieval. In *Proceedings of the 1995 ACM Multimedia Conference*, pages 305–313, San Francisco, CA, November 1995.

[17] C.E. Jacobs, A. Finkelstein, and D.H. Salesin. Fast mulit-resolution image querying. In *Proceedings of the Computer Graphics Conference*, pages 277–286, Los Angeles, CA, August 1995.

[18] H.V. Jagadish. A retrieval technique for similar shape. In *Proceedings of the ACM SIGMOD Conference*, pages 208–217, May 1991.

[19] K.F. Jea and Y.C. Lee. Building efficient and flexible feature-based indexes. *Information Systems*, 16(6):653–662, 1990.

[20] T. Kato. Database architecture for content-based image retrieval. In *SPIE Proceedings of the International Society for Optical Engineering*, pages 112–123, San Jose, CA, 1992.

[21] T. Kato, T. Kurita, and H. Shimogaki. Intelligent visual interaction with image database systems – toward the multimedia personal interface. *Journal of Information Processing*, 14(2):134–143, 1991.

[22] P.M. Kelly, M. Cannon, and D.R. Hush. Query by image example: The comparison algorithm for navigating digital image databases (candid) approach. In *SPIE Proceedings of the Storage and Retrieval for Still Images and Video Databases III*, pages 238–249, February 1995.

[23] A. Kitamoto, C. Zhou, and M. Takagi. Similarity retrieval of noaa satellite imagery by graph matching. In *SPIE Proceedings of the Storage and Retrieval for Still Images and Video Databases I*, pages 60–73, February 1993.

[24] D.E. Knuth and L.M. Wegner, editors. *Visual Database Systems, II.* North-Holland, 1992.

[25] F. Korn, C. Faloutsos N. Sidiropoulos, E. Siegel, and Z. Protopapas. Fast nearest neighbor search in medical image databases. In *Proceedings of the 22th VLDB Conference*, pages 215–226, Mumbai, India, September 1996.

[26] H. Lu, B.C. Ooi, and K.L. Tan. Efficient image retrieval by color contents. In *Proceedings of the 1994 International Conference on Applications of Databases*, pages 95–108, Vadstena, Sweden, June 1994.

[27] A. Nagasaka and Y. Tanaka. Automatic video indexing and full-video search for objects. In *Visual Database Systems, II, IFIP*, pages 113–127, 1992.

[28] G. Nagy. Image databases. *Image and Vision Computing*, 3(3):111–117, 1985.

[29] W. Niblack, R. Barber, W. Equitz, M. Flicker, E. Glasman, D. Petkovic, P. Yanker, and C. Faloutsos. The QBIC project: Query images by content using color, texture and shape. In *SPIE V1908*, 1993.

[30] V.E. Ogle and M. Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE Compuer*, 28(9):40–48, September 1995.

[31] A. Pentland, R.W. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. Technical Report 255, MIT Media Lab Perceptual Computing, 1993.

[32] E.G.M. Petrakis and C. Faloutsos. Similarity searching in large image databases. Technical Report CS-TR-3388, University of Maryland Institute for Advanced Computer Studies Dept of Computer Science, University of Maryland, 1994.

[33] W.K. Pratt. *Digital Image Processing.* John-Wiley, 2nd edition, 1991.

[34] F. Rabitti and P. Savino. Image query processing based on multi-level signatures. In *Proceedings of the IR Conference*, pages 305–314, 1991.

[35] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.

[36] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison Wesley, 1989.

[37] R. Shann, D. Davis, J. Oakley, and F. White. Detection and characterization of carboniferous foraminifera for content-based retrieval from an image database. In *SPIE Proceedings of the Storage and Retrieval for Still Images and Video Databases I*, pages 188–197, February 1993.

[38] M. Shibata and S. Inoue. Associative retrieval method for image database. *Transactions of the Institute of Electronics, Information and Communication Engineers*, (D-II, J73D-II):526–534, 1990.

[39] J.R. Smith and S-F Chang. VisualSEEk: A fully automated content-based image query system. In *Proceedings of the 1996 ACM Multimedia Conference*, pages 87–98, Boston, MA, November 1996.

[40] S.W. Smoliar and H.J. Zhang. Content-based video indexing and retrieval. *IEEE Multimedia*, 1(2):62–72, 1994.

[41] P.L. Stanchev, A.W.M. Smeulders, and F.C.A. Groen. An approach to image indexing of documents. In *Visual Database Systems, II, IFIP*, pages 63–77, 1992.

[42] M.J. Swain. Interactive indexing into image database. In *SPIE V1908*, 1993.

[43] M.J. Swain and D.H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[44] K. Tanabe and J. Ohya. A similarity retrieval method for line drawing image database. *Progress in Image Analysis and Processing*, 1989.

[45] A. Treisman and R. Paterson. A feature integration theory of attention. *Cognit. Pyschol.*, 12:97–136, 1980.

[46] J. Zobel, A. Moffat, and R. Sacks-Davis. An efficient indexing technique for full-text database systems. In *Proceedings of the 18th VLDB Conference*, pages 352–362, Vancouver, Canada, August 1992.