Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 40 (2007) 1711–1721
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/pr

# Visual guided navigation for image retrieval

Guoping Qiu*, Jeremy Morris, Xunli Fan

*School of Computer Science, University of Nottingham, Jubilee Campus, Nottingham NG8 1BB, UK*

Received 12 April 2006; accepted 27 September 2006

**Abstract**

In this work, we are interested in technologies that will allow users to actively browse and navigate large image databases and to retrieve images through interactive fast browsing and navigation. The development of a browsing/navigation-based image retrieval system has at least two challenges. The first is that the system's graphical user interface (GUI) should intuitively reflect the distribution of the images in the database in order to provide the users with a mental picture of the database content and a sense of orientation during the course of browsing/navigation. The second is that it has to be fast and responsive, and be able to respond to users actions at an interactive speed in order to engage the users. We have developed a method that attempts to address these challenges of a browsing/navigation based image retrieval systems. The unique feature of the method is that we take an integrated approach to the design of the browsing/navigation GUI and the indexing and organization of the images in the database. The GUI is tightly coupled with the algorithms that run in the background. The visual cues of the GUI are logically linked with various parts of the repository (image clusters of various particular visual themes) thus providing intuitive correspondences between the GUI and the database contents. In the backend, the images are organized into a binary tree data structure using a sequential maximal information coding algorithm and each image is indexed by an $n$-bit binary index thus making response to users' action very fast. We present experimental results to demonstrate the usefulness of our method both as a pre-filtering tool and for developing browsing/navigation systems for fast image retrieval from large image databases.
© 2006 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

*Keywords:* Image database; Image retrieval; Browsing/navigation; Entropy; Information theory; Color

## 1. Introduction

Managing large image database and providing effective tools for users to quickly find image items they are looking for is still a very challenging problem. In the past decade or so, the content-based image indexing and retrieval (CBIR) paradigm has dominated the research community, e.g. Refs. [1–6]. There are a number of intrinsic weaknesses associated with the traditional CBIR paradigm, which have hindered progress. First, the objectives of CBIR are ill defined. Although the idea of using visual examples to find similar images is sound and intuitive, it is problematic in practice. It is not clear in what circumstances/application scenarios users would want/prefer to search images by example. The definition of CBIR is too broad and vague; retrieval by example can mean different things to different users and in different applications. Even finding a starting query example can be problematic because some visual forms are intrinsically hard to describe precisely. Second, current state of the art technologies are not yet mature enough to realize the ideal of CBIR. In particular, it is difficult to compute image similarity measures that match the perceptual differences between images. Automatically retrieved images are often not what the users expected, hence there is a gap between the retrieval results and users' expectation. Third and fundamentally, the CBIR paradigm puts the user in a passive position in the sense that retrieval results are largely determined by the computational algorithms; the user cannot actively control the retrieval results. The immaturity of the enabling computational algorithms has only made the task even more difficult.

Recent trends in CBIR have been to introduce browsing, navigation and relevant feedback facilities to enable users to interact with the retrieval system and to engage the users, examples include Refs. [7–12]. The advantages of a

* Corresponding author. Tel.: +44 115 8466507; fax: +44 115 9514254.
*E-mail address:* qiu@cs.nott.ac.uk (G. Qiu).

0031-3203/$30.00 © 2006 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2006.09.020

browsing-based approach are that it allows the user to take a more active initiative in the retrieval process thus enhancing their experiences in dealing with image retrieval. In our experience, it seems that a browsing-based paradigm also more naturally matches users' behaviors. Informal experiments we conducted have showed that users tended to like browsing/navigating through the database when looking for a particular item. Of course, since image retrieval is such a complicated and subjective process, a comprehensive system will necessarily require technologies for CBIR, browsing/navigation and relevant feedback etc. to work together. In this work, we attempt to develop a browsing/navigation-based tool for image retrieval.

To design and implement an effective browsing/navigation tool for image retrieval, there are several challenges. Firstly, it will require an appropriately designed graphical user interface (GUI). The GUI should provide cues and visualization facilities to "guide" users to navigate through the database. This means that the design of the GUI and the way the image items are indexed/organized have to be coordinated. In other words, the GUI should intuitively reflect the distribution of the images in the database thus providing the users with a mental picture of the database content and to provide the users with a sense of orientation during the browsing/navigation process. Secondly, it has to be fast, responsive and instant. The tool should be able to respond to users action at an interactive speed, otherwise it will not be able to engage the users and users may loose interest thus reducing the effectiveness of the tool. This means that indexing and organization of the database content have to be designed in such a way that responses to user requests can be computed very efficiently and fast.

In this paper, we present a method that attempts to address aforementioned challenges of a browsing/navigation-based image retrieval system. The unique feature of the method is that we take an integrated approach to the design of the browsing/navigation GUI and the indexing and organization of the images in the database. The GUI is integrated with the algorithms that run in the background. The visual cues of the GUI are logically linked with various parts of the repository (image clusters of various particular visual themes) thus providing intuitive correspondences between the GUI and the database contents. In the backend, the images are organized into a binary tree data structure. Each image is indexed by an $n$-bit binary index. Image retrieval is performed by traversing through the binary tree data structure using the $n$-bit binary keys thus making response to users' action very fast.[1]

The organization of the paper is as follows. In Section 2, we introduce a maximal information indexing method to index and organize the images in the database. In Section 3, we exploit the maximal information indexing method to design a GUI which naturally links images and indexing features with visual cues in the GUI. Section 4 presents our experimental prototype and experimental results. Section 5 concludes the presentation. Preliminary versions of this work have been published in Refs. [25,26].

## 2. Maximal information fast image indexing

In traditional CBIR, an image is represented by some low-level feature vectors. Image retrieval is based on comparing the querying image's low-level features with those of model images in the database. These features are normally of very high dimension and therefore comparing the images is computationally a very expensive process. Not only is distance calculation expensive, but also image search is very time consuming as well because a querying image has to be compared with all the images in the database (exhaustive/sequential search). Previous fast approaches such as Ref. [8] uses tree data structure to reduce the search effort but still requires the computation of high dimensional distances for traversing the tree.

Ideally, we want to have a simple data structure and efficient indexing scheme. Each image in the database should have a key and image querying should only amount to retrieving images in the database that have the same or similar keys as the querying image without the need to compute low level distance measures. When it is necessary to use high dimensional feature vectors, it should ideally only perform on a (much smaller) sub group of images that is most likely to contain the targets instead of searching the whole database.

Perhaps a good starting point for developing computer systems to manage large image databases is to ask the question: how would humans do it? After all, humans are very good at this task. We have done a very informal psychovisual experiment to try to answer this question. We found that the most common approach taken by the participating subjects is to first divide the images into small number of groups, each containing images with easily describable common visual themes. Then each group is divided recursively into smaller sub groups, again with each sub group containing easily describable common visual themes. We also found that color tones are the most apparent visual themes to come out of the grouping. Such a hierarchical division of images according to certain visual themes is schematically illustrated in Fig. 1.

Although we are not sure exactly how human might have been "programmed" to perform tasks as this, we find the approach illustrated in Fig. 1 interesting and illuminating which might offer us useful guidance to design computer programs to manage large image database. From a pragmatic point of view, such a scheme makes engineering sense also because it is generally a good idea to divide a large task into smaller, more manageable sub tasks and tackle each sub tasks in turn according to the unique nature of these smaller

[1] Strictly speaking, most CBIR systems do not actually index images, but rather store metadata of the image content such as color histogram, texture descriptor, etc. Image retrieval is normally performed by calculating the distance measures of these metadata features. Such approach may be more accurate but demand more storage space and higher computational costs.

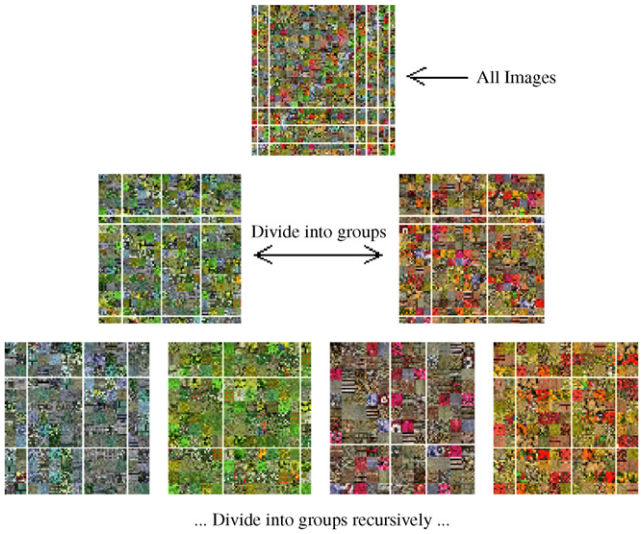


Fig. 1. A schematic illustration of one of the possible approaches to organizing large image collections by recursively dividing large collections into smaller groupings according certain visual themes.

sub problems. How can computer divide a large image repository into smaller groups that make visual sense? Note that it is important to emphasise that each grouping should have an intuitive visual theme that can be easily grasped by the users. Because visual experiences are subjective, this is one of the most challenging problems in computational vision. From the computer's point of view, pixel values are all it has got and therefore it will have no choice but to use these low-level representations. The question is, how?

There have been many studies that support the hypothesis that biological sensory neurons are adapted to the statistical signal to which they are exposed [13]. Although such hypothesis is still not proven, we believe it helps our thinking in terms of relating low level visual features to high level visual phenomenon. Therefore, we believe any meaningful visual theme division should be based on statistics of the visual stimulus occur in natural scenes. A number of psychophysical studies have shown that animals are sensitive to both first and second-order properties of the visual input. The statistics that have been studied include intensity statistics, color statistics, spatial correlation, higher order statistics including principal component analysis (PCA) and independent component analysis (ICA), and spatial—time statistics [13].

How do these (low-level) statistics relate to visual phenomenon? This is a difficult and not very well understood problem. Equally difficult and crucially important from building computer systems point of view is the question—how can we relate the (low-level) visual statistics to (high-level) visual phenomenon *numerically*? There have been much cognitive neuroscience studies on the topic of scene categorization performed by human and their relation to low-level scene statistics [13–19]. However, high-level scene categories, such as the so-called superordinate (e.g., man-made/natural), basic (e.g., city/highway) and subordinate (e.g., particular example of a city), are still very difficult to represent and distinguish numerically. These cognitive neuroscience studies may eventually be exploited in managing large image database for image retrieval, but in the current work we take a more intuitive and simpler route. We attempt to categorize scenes according to their color impressions. This is because many studies have shown that color is an important and useful cue in recognizing images, and from a computational point of view, this is more manageable. Although the use of color has been extensively studied in the image database literature [1–6], our use of color information in this paper is rather unique. We use color to unify image indexing and browsing graphical user interface design. Furthermore, our use of color can be seen as a design philosophy for image retrieval systems. The design principles of our work may be extended to include more visual cues.

In developing our solution, we are reminded of the challenges, which are that we have to seek computationally fast and organizationally simple indexing methods, and simultaneously, such simple methods have to be easily exploited to design an effective and intuitive browsing GUI. Ordinarily, color histogram can be used to characterize the color distributions of images, however, we decided to go against the idea of using color histogram for two reasons. Firstly, color histograms are high-dimensional vectors that will inevitably involve computationally expensive processes such as computing the $L_1$ or $L_2$ norms of the (high dimensional) histograms. Secondly and equally importantly is that, it is not easy to tell the apparent "visual themes" of an image from its high dimensional color histogram. That is, from an image's $k$-bin color histogram, we cannot easily describe the color impressions of the image. What we want is something much simpler, for example, a single number that can be used to describe the rough visual impression of the image, which users can grasp intuitively. We believe this is possible. After all, it has long been established [23] and supported by more recent comprehensive studies [24] that there exists a cross-linguistic universal inventory of exactly 11 basic color categories from which the 11 or fewer basic color terms of any given languages are always drawn. In English, these 11 color terms include, eight chromatic terms, red, yellow, green, blue, purple, brown, orange, pink; and three achromatic terms, black, white and gray. So how can we use the color statistics simply and in a visually apparent, intuitive, and meaningful way to organize large image database for retrieval?

A well-known color vision theory is the opponent color theory [20], which suggests that there are three visual pathways in the human color vision system. One pathway is sensitive mainly to light–dark variations; this pathway has the best spatial resolution. The other two pathways are sensitive to red–green and blue–yellow variation (the opponent channels). The blue–yellow pathway has the worst spatial resolution. In opponent-color representation, the spatial sharpness

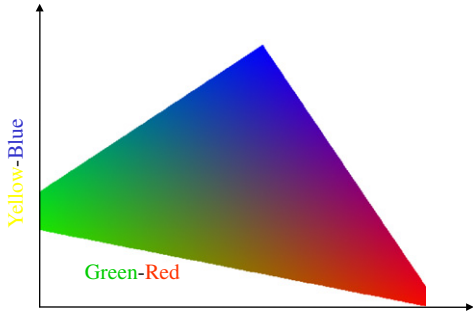


Fig. 2. Color appearances in the red–green (*rg*) and yellow–blue (*yb*) chromatic space (the light dark component was fixed to a constant).

of a color image depends mainly on the sharpness of the light dark component of the images and very little on the structure of the opponent-color image components. There is evidence to suggest that different visual pathways process color and spatial pattern in the human visual system. These human vision theories suggest that the colors of an image are mainly encoded in the opponent color signals whilst the spatial patterns or textures are mostly encoded in the light–dark component. In the current work, we concentrate our effort on the use of colors and therefore only use the opponent color components to index the images. To add texture we can add the light–dark component of the image follow a similar principle (we will discuss this extension later in the paper).

Assuming that the original data is in RGB color space, the red–green (*rg*) and blue–yellow (*by*) components can be calculated as (we have experimented with other forms of chromaticity calculation formulae, results are similar)

$$rg = R - G, \quad by = \tfrac{1}{2}(R + G) - B. \tag{1}$$

Fig. 2 plots the chromaticity diagram in the *rg*–*by* space. As we move from the origin along the *rg*-axis the colors changes from green to red, whilst moving from the origin along the *by*-axis the colors change green/red to blue. Clearly, a certain value of *rg* or *by* has an intuitive correspondence with the color appearance, e.g., a large *rg* value indicate the red color and a large *by* value indicates the blue color, etc.

Our idea is to directly use the chromaticity diagram as our GUI and use the *rg* and *by* signals as the indexing features. We want to divide our image database into clusters, each having a particular visual theme corresponding to a color, or equivalently, a particular pair of *rg/by* values on the chromaticity diagram. In this way, we establish a simple and intuitive link amongst the (color) appearance of the image, the numerical representation of the image, as well as a region in the GUI. We will now first describe an algorithm for using the *rg/by* opponent components to cluster images and then describe in more detail how such an image clustering scheme can be combined with the chromatic diagram to design an image database browsing/navigation GUI.

There have been many research works studied sensory coding and natural scene statistics. The efficient coding hypothesis predicts that individual neurons should maximize information transmission [13,21]. Although we are not interested in the hypothesis itself, the information maximization coding principle may be "borrowed" for indexing large image database. Suppose we have a signal, $X$, to be coded by an encoder as $C(X)$ and the response of the encoder is limited within some bonded intervals. In order to convey the maximal information, it is straightforward to show that the distribution of the encoder response must be uniformly distributed within the bonded interval. According to Shannon information theory [22], information content is measured as entropy

$$H(X) = -\sum_{\forall X} P(X)\ln(P(X)). \tag{2}$$

$H(X)$ reaches maximum when $P(X)$ is a uniform distribution. Therefore, maximal information and maximum entropy (maxent) are equivalent.

To exploit the maximal information encoding principle for image indexing and retrievals turns out to be less straightforward than at first sight. In the human visual system, there could well be multiple channels that code the incoming sensory signal in parallel. However, in engineering practice, it is very difficult to compute information of multidimensional random variables because of the difficulty in estimating the probability density functions in high-dimensional spaces. To get round this computational difficulty, we simplify the model by assuming (not necessarily make biological sense) that the channels are coded independently in a sequential manner and each follows a maximal information principle. For any given image, each *rg/by* component in Eq. (1) is still of very high dimension (image height by image width). To simplify the problem further, we only model the first order statistics by considering pixels in each of the component independently, that is, we collapse each component into a sequence of scalar values. In this case, a histogram of the pixel distribution in each of the channels can be easily compiled. For a scalar random variable, constructing a maximal information encoder with limited alphabets becomes a simple task. All is needed is to ensure that the pixel populations distributed to each alphabet is identical. With such a simplified model, we can encode the components in Eq. (1) independently. Each component is then again treated as multiple instances of a scalar random variable. A scalar encoder that conveys the maximal information for each channels can then be easily constructed.

How the maximal information encoders such as those described above can be used for image database indexing and retrieval? Here is a scheme we have developed which has been found to work well. In our developed scheme, a binary tree data structure is constructed by encoding the *rg/by* components in Eq. (1) in a hierarchical order as illustrated in Fig. 3.

Starting from level 0, for a pre-selected component Ch(0) (each Ch($m$, $n$) represents either *rg* or *by*), we find a threshold value, $T(0)$, that cuts all pixels in this component from

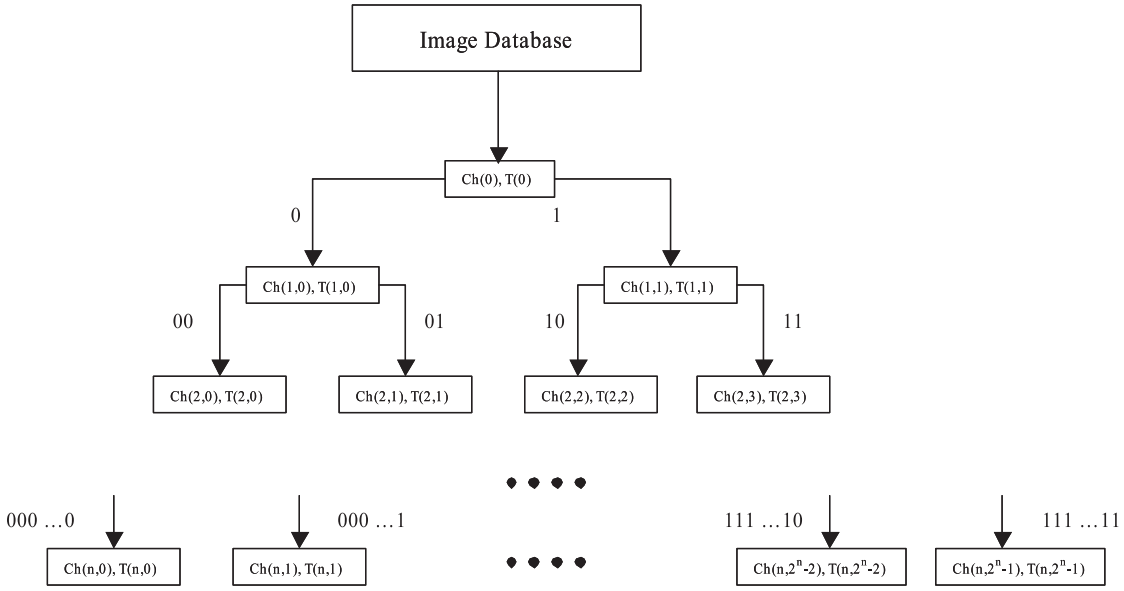


Fig. 3. Binary tree indexing of images based on maximal information sequential visual coding.

*all images* in the database into two equal population halves, this value is sometime known as the median of the pixels. It is important to note that at this initial node, all pixels from all images in the database participate in finding this median value. Note also that to code the component into two alphabets, the maximal information coding dictates that the pixels must be equally distributed into the two alphabets [22]. Once we have found the median value $T(0)$, we then divide the images in the database into two clusters, each placed in one of the two child nodes. Those images whose Ch(0) component have more pixels with values greater than $T(0)$ are classified to the right child node and those images have more pixels with values smaller than $T(0)$ are put into the left child node. The two child nodes of the root node form level 1 of the hierarchy. At level 1, images are indexed by a 1-bit key. All images in the left child node will be indexed by a 0-value and all images in the right child node will be indexed by a 1-value. The component used by the node and the median value of that component is stored with the tree.

For each non-leaf node in the tree, a pre-selected component (*rg* or *by*) and its median value computed from pixels from all images falling into that node, are again used to divide the images in this node into its left and right child nodes. For images classified into the left child node, a 0-value bit is appended to their key and for images classified into the right child node, a 1-value bit is appended to the images key. Again information about the component used in the node and its median value are stored with the tree. For an $n$-level tree so constructed, each image is indexed by an $n$-bit *binary key*. The keys between images in a parent node and its immediate two child nodes differ by one bit.

Once the tree has been constructed, new images can be easily put into the tree. All we have to do is to compare the median values of the various components of the image corresponding to those orders used in the tree to assign the image into appropriate nodes, and the image will be indexed with the keys of the nodes it belongs. The reasons we can do this is that we can assume that a large number of images have been used to construct the initial tree and by adding one extra image will not change the overall median pixel values much. Of course, if a lot more images are to be added to the database, the tree can be easily reconstructed by including the new images. Because only the median values of the pixels need to be found, the indexing process is computationally very simple.

The indexing scheme also enables querying by example images. Query-by-example follows the same procedures as that for adding an image to the database. Once the binary key of the query image has been computed, there is considerable flexibility in retrieval. We can either return all images in the leaf node the querying image falling into, or images at the non-leaf nodes of the tree the querying image belongs to (more returns).

Navigating through the tree (and therefore the database) is also easy. We can transverse the tree by going up and down the nodes. At any particular node, we could output the images at the node for visualization and then decide to see more images (by going to the parent node) or see fewer images (by going to one of the child nodes).

From storage's point of view, each image has only $n$-bit binary extra information to store. This is in contrast to traditional CBIR [3,4] where very high-dimensional metadata such as histogram, texture descriptor etc. have to be stored. For the tree itself, we only need to store information of the component used in each non-leaf node and the component's median value of all (training) images. Therefore, the overhead of our system is very small. We are not aware of any CBIR image database management system has the efficiency

of this method. Because we actually index our images, which makes image retrieval very efficient in our scheme. Unlike many existing methods for CBIR, where much side information in the form of image content descriptors, such as color histograms, will have to be stored, and image retrieval is done by first computing the distance measures of these image descriptors and then linearly searching the image database, our method needs only to make $n$ ($n =$ levels of the tree) scalar comparison operations to obtain the key and $n$-bit binary comparison to retrieval the images.

Of course, since our indexing scheme is simple, it may not have the accuracy of the traditional CBIR technology (however, due to the subjective nature of the problem, accuracy may be itself very difficult to define and it is not yet clear whether a more complicated system will be actually better). There might be a trade off between simplistic and accuracy. We believe our scheme is useful to pre-filter a large image database and to narrow down the search scope so that more accurate but computationally more demanding retrieval technologies can be applied to a smaller number of images. As mentioned in the introduction, human users tend to have the habit of actively browsing the database. One of the advantages of our new indexing technique is that it can be easily exploited to design intuitive and easy to use GUI for browsing and navigating image database. In the next section, we will describe a GUI design that exploits our indexing scheme for browsing/navigating image databases and for fast image retrieval.

## 3. Visual guided browsing/navigation

The indexing scheme described in Section 2 renders the red–green/blue–yellow chromaticity diagram readily usable as a graphical user interface for browsing/navigating image database. The process of constructing the indexing tree in Fig. 3 in fact partitions the chromaticity diagram as illustrated in Fig. 4.

The partitioned regions of the chromaticity diagram and the nodes of the tree in Fig. 3 have a one to one correspondence. At the root node, the chromaticity diagram is partitioned into two halves; each associated with the two child nodes at level 1. The images classified into the left child node and the left half of the chromaticity diagram can be assigned one key value and the images classified into the right child node and the right half of the chromaticity diagram can be assigned another key value. As we construct the tree into more levels, the chromaticity diagram is being recursively partitioned into smaller regions, each associated with a tree node and the images distributed into the node. A region in the chromaticity diagram and the images classified into its corresponding node are assigned the same indexing key. In this way, each region of the chromaticity diagram is associated with a key, which is coupled with images with the same key. Importantly, if an image and an interface region share the same key, they will share similar visual appearance (color in this case) because when an image and an interface region are assigned the same key, the image will have majority of its pixel's chromaticity values falling within that region, therefore it is reasonable to assume the overall visual (color) impression of the image will be similar to its corresponding interface region. This is useful because it can provide a sense of orientation and visually guide browsing/navigation process.

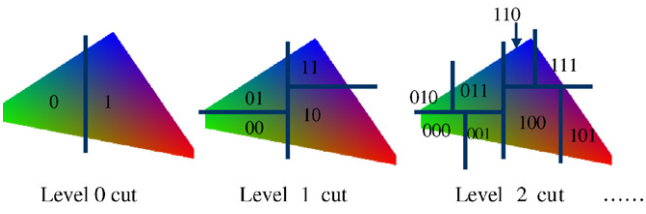


Fig. 4. Constructing the indexing tree recursively partitions the chromaticity diagram into smaller regions. Each partitioned region share the same key with images distributed into the tree node with the same key. The images having the same key as a partitioned region in the diagram share similar color appearances.

Based on these associations between the chromaticity diagram and the image indexing/clustering scheme in Section 2, we can design a GUI for navigating and browsing images in the database, as illustrated in Fig. 5.

## 4. Experimental results

In the first experiment, we tested the indexing method's retrieval performances. There were 60,000 color images (from Corel photo CD collection) in our database, which have been put into a tree constructed using the method described in Section 2. In constructing the tree, we simply alternated the *rg* and *by* channels from one level to the next and all nodes on the same level used the same channel for indexing. We used two testing image sets to evaluate the performances. The first testing set consists of 168 pairs of images. Examples of these pairs are shown in Fig. 6. We put the 168 pairs of images in the tree constructed using the 60,000 images. The purpose is to see whether each pair of images will have the same key at various level of the tree. If so, then imagine using one of the pairs as querying image, it will directly find the image cluster of the corresponding target image. If the cluster is of small size, then finding the target should be made much easier.

Table 1 shows the number of image pairs staying together at various leaf nodes for different depths of the indexing tree. The longer are the keys, the smaller are the sizes of image clusters in the leaf nodes, hence the chances of the same pair falling into the same node are decreased. It is seen that even using an 11-bit key (on average, a leaf node will contain a cluster of about 30 images and of course the numbers of images in the nodes are unevenly distributed), 111 out of 168 pairs managed to stay together in various leaf nodes. Reducing the key length to 9 bits (on average a leaf node

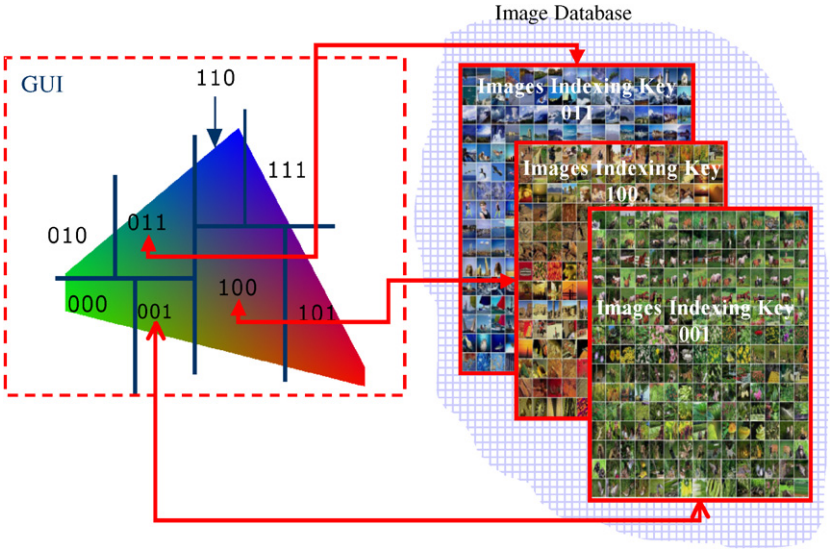


Fig. 5. Using the chromaticity diagram directly in the graphical user interface. Each region and images shared the same key will have similar color appearances, hence the chromaticity diagram can guide the browsing of the database. By placing the mouse cursor on a particular area (color) in the chromaticity diagram, the action is automatically associated with the images having similar color appearances, and by clicking a mouse button, images having the same key as the color pointed at by the mouse can be quickly returned to the user.


Fig. 6. Testing image set 1. For an image in set A, there is one corresponding image in set B. The corresponding images form a pair of querying and target images. The purpose is to test if the same pair of image will be assigned the same key at various level of the tree.

cluster will contain about 117 images), there were 162 out of 168 pairs managed to stay together at various leaf nodes of the tree.

These results indicated that the method could be very useful. For example, if we were to use one of the pairs as querying example and the other image of the pair was in the database and we wanted to find it. If we used 9-bit indexing keys to organize our database using the indexing method of Section 2, then in 162 out of 168 queries, on average, we can quickly narrow down the search to 117 images, instead of having to search through 60,000 images. Therefore, our method can be first used as a pre-filtering technique to reducing the search range, and then computationally more demanding (perhaps more accurate) methods such as traditional CBIR techniques can be used to search a smaller subset that is most likely to contain the target images. Furthermore, manually browsing through a subset of 117 images also becomes manageable.

Table 1
Pairs of images stayed together at various leaf nodes of the tree (total 168 pairs)

| Key length | 11 bits | 10 bits | 9 bits |
|---|---|---|---|
| # pairs stay together at various leaf nodes | 111 | 156 | 162 |

In another test, we used 120 classes of color texture images. Each class consists of six similar images. Examples of these are shown in Fig. 7. Again, we put these 720 images ($6 \times 120$) into the indexing tree constructed using the 60,000 images and the method described in Section 2. This time, the purpose is to test how many of the classes will have all of their member images stay together at the various leaf nodes of the tree of different depths. Again, we can imagine that using one of the images from a particular class to query the image database with the aim of finding the other five images belong to the same class. If all images of the same class stay together at a particular leaf node, then we need only to search images belong to that leaf node. Results for various tree depths are shown in Table 2. It is seen that, when using a 9-bit key to index the images, each and every class's six images all stay together at a particular leaf node. That is, in each query (out of a total 720 queries), on average, we need only to search 117 images to find all five targets from the 60,000-image database. These results further demonstrated the usefulness of our indexing method in narrowing down the search range in preparation for more accurate but more computationally demanding retrieval or even for manually browsing a subset to find the targets.

To validate the usefulness of our technique for browsing/navigation as well as image retrieval, we have implemented the indexing scheme of Section 2 and the image database browsing/navigation GUI schematic of Fig. 5 in a prototype system. Fig. 8 shows screenshots of our system using the chromaticity diagram to navigate/browse image database. The indexing scheme of Section 2 can be used to retrieve image based on example. For a given querying example, we can easily compute the example image's indexing key by comparing the *rg/by* channels median values with those stored in the indexing tree. With the indexing key, we can retrieve those images having the same key as the querying example. As we have seen in the first experiment, we can indeed narrow down the search range dramatically. We have implemented this functionality into our prototype as shown in Fig. 9.

The interface of the system was implemented as a proof of concept for testing the technology proposed in this paper, therefore, the features are by no means complete or optimal. Briefly, the system allows the users to do the following. It has a chromaticity diagram, each color is assigned an indexing key as illustrated in Fig. 4. Implicitly, these keys correspond to the nodes of the indexing tree (hence images

Table 2
Number of classes with all six sub images stay together at various leaf nodes of the tree

| Key length | 11 bits | 10 bits | 9 bits |
|---|---|---|---|
| # classes with all six images stay together at various leaf nodes | 92 | 115 | 120 |

Fig. 7. Testing image set 2. Each row is the six images belonging to the same class and there were a total of 120 classes.

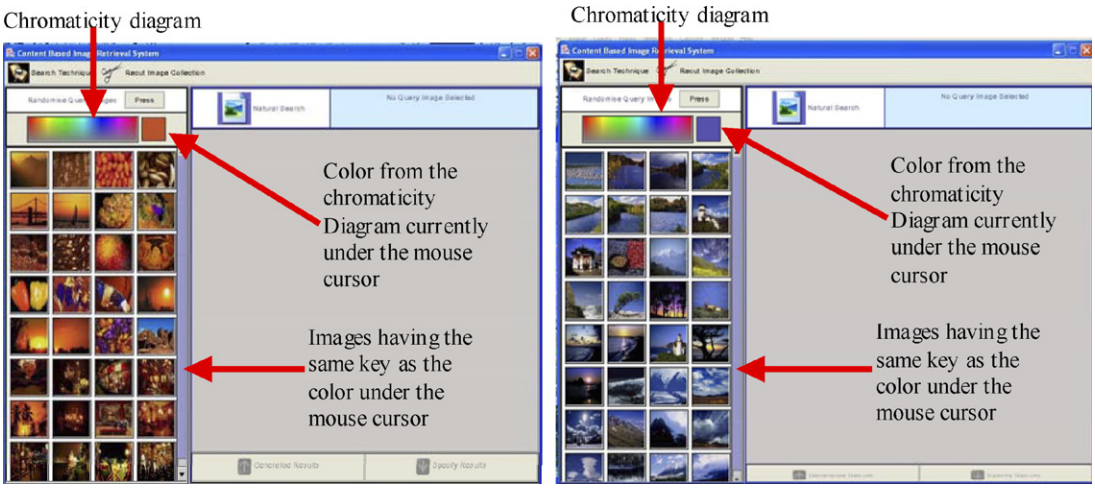


Fig. 8. Image database navigation/browsing. User places the mouse cursor in an area in the chromaticity diagram where the color corresponds with the color appearances of the images the user have in mind. By clicking a mouse button, all image in the database having the same key as the color currently under the mouse cursor will be displayed instantly on the display panel. User can navigate through the image database by moving the mouse cursor around the chromaticity diagram and browsing images with a color theme similar to that pointed to by the mouse cursor. The chromaticity diagram is served as a mirror to the image database because any color region on the diagram is associated with images having similar color appearances. Therefore the chromaticity diagram acted as a visual guide for browsing/navigating the database.


Fig. 9. Query by example. By pressing a button, a random set of initial images are displayed on the left display panel from where a querying example can be selected. We can also use the chromaticity diagram to retrieve an initial set of example images. Once an example image is selected, all images in the leaf node having the same index as the example will be displayed on the right display panel. There are also two navigation buttons for traversing up and down the tree. The figure on the right shows images at a parent node two levels up from those shown on the left figure (by pressing the up button twice).

in the database). By selecting a color from the chromatic diagram, images in the leaf node having the same key as the color will be displayed instantly. The system also allow query by example. The query example can be selected randomly from the database, or from a set of images returned by selecting a color from the chromaticity diagram. Once a querying example is selected, all images in the leaf node having the same key as the example image will be displayed instantly. There are also two navigation buttons allow user to traverse up and down the tree to allow more or fewer images to be displayed.

Ultimately, we wish to use our method to develop a useful tool to help users to manage their image database. One way to test the usefulness of a tool is to see if it can *actually* help users to find image items quickly. To do this, we have performed a subjective experiment using our prototype system.

This time, we used 10,000 images (a subset of the 60,000 image used in the first experiment) in our prototype system. From these 10,000 images, we randomly selected 10 images as shown in Fig. 10.

We then printed these images in hardcopies and gave these hardcopies to five users (all computer science


Fig. 10. Ten images randomly picked from a 10,000-image database used in the subjective experiment.

Table 3
Averaged time taken by 5 users to find 10 images printed on paper from a 10,000-image database

| Image number | Searching manually | Searching using our prototype |
|---|---|---|
| 1 | 7 min 3 s | 1 min 6 s |
| 2 | 7 min 43 s | 3 min 59 s |
| 3 | 1 min 34 s | 44 s |
| 4 | 25 s | 5 min 36 s |
| 5 | 1 min 54 s | 1 min 43 s |
| 6 | 5 min 41 s | 2 min 6 s |
| 7 | 3 min 38 s | 2 min 21 s |
| 8 | 11 min 29 s | 6 min 35 s |
| 9 | 2 min 39 s | 2 min 19 s |
| 10 | 5 min 3 s | 9 s |

undergraduate students without any knowledge of our system and our technique). We asked these five users to find the 10 images printed on paper from the database. The user could find the image using two methods. Firstly they would find the image manually by trawling through the directories on the system until the image was found. The tool used for manual search was MS Windows Explorer. The images in the system were organized into subdirectories each contained 100 images. Once a directory is selected, the thumbnail images will be displayed instantly without delay on a latest Pentium PC with 512MB memory. Secondly they would use our prototype system to find the same images. We timed how long it took a user to find the images. We then compared the times taken to find the printed images using both methods to find out whether our system shortened retrieval times and therefore was useful to the user. Results are shown in Table 3.

The times in Table 3 suggest that the systems and techniques developed in this paper is a useful aid in which to retrieve images. Out of the 10 query images, only image 4 was retrieved faster using a manual search technique. Image 4 was situated in an early searched directory, which is the reason that it was retrieved so quickly. We must accept that if the user is fortunate enough to find the image quickly by manually trawling through directories, then the retrieval time is likely to be faster than by using the prototype system. But in general our system has performed better than a manual search and on average has cut the retrieval time by half.

Because of the subjective nature of image retrieval, accurate system evaluation is difficult. However, all users who have tested our prototype system have agreed that the usability of the system is adequate. They have all been able to search for query images effectively and seem to enjoy using the system. In 50 rounds (5 users $\times$ 10 images) of image retrieval exercises, users were able to use a primitive tool developed based on our technology to half the average time taken to find a particular image.

Because the technique was based on color only, as expected, it has its drawbacks. As the algorithm calculates the median value from all pixels in an image, images with many colors have not performed so well as an image that has one predominant color. Using our prototype system, the two images which took the longest time to retrieve were images 4 and 8. These two images are both made up of three predominant colors as opposed to the other images which mainly have one predominant color. Despite retrieval time being longer for mixed color images, our system still retrieved those images in reasonable time. The system does have occasional incongruent images but generally the groupings are accurate and representational. In a sense, our technique and system trades simplicity for accuracy. Whether a computationally more demanding but more accurate indexing technique will be better and enables faster image retrieval than a simple and fast but less accurate method is something of interest for future research.

It is worth mentioning that the purpose here is not to compete with the most comprehensive and state of the art image retrieval systems because these will inevitably depending on the software utilities as well as image indexing methodologies. Rather, we want to show how simple and yet effective our proposed technique is. A web-based system that implements the same techniques and ideas of this paper can be viewed online at: http://www.theimagerepository.co.uk. This web-based system was implemented by Ben Bedwell in his BSc(Hon) Computer science final project. This shows

yet again the simplicity of the technique which can be easily implemented by even people without much knowledge of image processing.

## 5. Conclusions and future work

In this paper, we have developed a simple and very efficient technique to organize large image database. The advantage of the method is that indexing and retrieval can be computed very fast and efficiently. The method requires very little storage overheads. An important feature of the technique is that it readily renders itself for developing intuitive and easy to use graphical user interface for browsing image database. We have conducted two sets of experiments to evaluate the developed technology. The first experiment tested the usefulness of the technique as pre-filtering method for narrowing down the search range in preparation for more accurate and computationally more demanding retrieval or for manually browsing a subset of images likely to contain the target images. Results have demonstrated quite convincingly that our method as a pre-filtering method is useful. In the second experiment, we integrated our technology into a primitive prototype system and used real users to test the usefulness of our technology. In a small scale test, results indicated that the technique can be employed to almost half the image search time from a 10,000 image database, again demonstrated the usefulness of our technique.

The technique can be extended in the future. One obvious direction is to include visual feature other than color, e.g., texture, or shape and size of objects contained in the images. For example, we could easily compute a Wavelet transform on the dark-light component of the images and follow the same information maximization principle of Section 2 to partition the database according to their Wavelet coefficients. Intuitively, larger Wavelet coefficients indicate rougher surface and smaller Wavelet coefficients indicate smoother surface. However, as more features are added, the indexing structure may become more complicated, what is more, it may not be as easy to develop GUIs that have a direct and intuitive correspondence with the database structure which will allow users to form a mental picture of the database to intuitively browsing/navigating through the database. Our future work will investigate the incorporation of more indexing features such that image groupings will be more accurate, but simultaneously, we want to maintain the design philosophy of this paper, i.e., fast indexing, small overheads, instance response, intuitive GUIs and tight coupling of the GUIs and the indexing data structure. The question of whether to use fewer indexing feature, hence simpler indexing structure and more intuitive GUI, or to use more indexing features, hence more complicated indexing structure and more cluttered GUI, will lead to better image retrieval systems needs systematical investigation. How to combine our current method with existing computationally and storage more demanding CBIR technology and/or relevant feedback is also a direction we will study in the future.

## References


[1] W. Niblack, et al., Querying images by content using color, texture and shape, Proc. SPIE 1908 (1993) 173–187.

[2] M. Swain, D. Ballard, Color indexing, Int. J. Comput. Vision 7 (1991) 11–32.

[3] Y. Rui, T.S. Huang, S.F. Chang, Image retrieval: current techniques, promising directions, and open issues, J. Visual Commun. Image Representation 10 (1999) 39–62.

[4] A.W.M. Smeulders, et al., Content-based image retrieval at the end of the early years, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 1349–1380.

[5] B.S. Manjunath, W.Y. Ma, Texture features for browsing and retrieval of image data, IEEE Trans. Pattern Anal. Mach. Intell. 18 (1996) 837–842.

[6] C. Carson, S. Belongie, H. Greenspan, J. Malik, Blobworld: image segmentation using expectation-maximization and its application to image querying, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2002) 1026–1038.

[7] J. Vendrig, et al., Filter image browsing: interactive image retrieval by using database overviews, Multimedia Tools Appl. 15 (2001) 83–103.

[8] J.Y. Chen, C.A. Bouman, J.C. Dalton, Hierarchical browsing and search of large image databases, IEEE Trans. Image Process. 9 (3) (2000) 442–455.

[9] J. Laaksonen, et al., Self-organizing maps as a relevant feedback technique in content-based image retrieval, Pattern Anal. Appl. 4 (2001) 140–152.

[10] Y. Rubner, Perceptual metrics for image database navigation, Ph.D. Thesis, Stanford University, May 1999.

[11] S. Santini, R. Jain, Integrated browsing and querying for image databases, IEEE MultiMedia 7 (3) (2000) 26–39.

[12] Int. J. Comput. Vision 56 (1–2) (2004) (Special Issue on Content-based Image Retrieval).

[13] E. Simoncelli, B. Olshausen, Natural image statistics and neural representation, Annu. Rev. Neurosci. 24 (2001).

[14] M. McCotter, F. Gosselin, P. Sowden, P. Schyns, The use of visual information in natural scenes, Vis. Cogn. 12 ( 6) (2005) 938–953.

[15] A. Torralba, A. Oliva, Statistics of natural image categories, Network: Comput. Neural Syst. 14 (2003) 391–412.

[16] E. Rosch, et al., Basic objects in natural categories, Cognitive Psychol. 8 (1976) 384–439.

[17] B. Tversky, K. Hemenway, Categories of environmental scenes, Cognitive Psychol. 15 (1983) 121–149.

[18] C.G. Gross, Coding for visual categories in the human brain, Nature Neuralsci. 3 (2000) 855–856.

[19] E. Miller, et al., Neural correlates of categories and concepts, Current Opin. Neurobiol. 13 (2003) 198–203.

[20] P.K. Kaiser, R.M. Boynton, Human Color Vision, Optical Society of America, Washington, DC, 1996.

[21] S.B. Laughlin, A simple coding procedure enhances a neuron's information capacity, Z. Naturforsch 36c (1981) 910–912.

[22] T.M. Cover, Elements of Information Theory, Wiley, New York, 1991.

[23] B. Berlin, P. Kay, Basic Color Terms, University of California Press, Berkeley and Los Angeles, 1969.

[24] P. Kay, T. Regier, Resolving the question of color naming universals, Proc. Natl. Acad. Sci. 100 (15) (2003) 9085–9089.

[25] G. Qiu, L. Ye, X. Feng, Fast image indexing and visual guided browsing, CBMI 2003, Third International Workshop on Content-Based Multimedia Indexing, September 22–24, 2003 IRISA, Rennes, France.

[26] G. Qiu, J. Morris, X. Fan, From sensory coding to scene classification, MMSP2004, IEEE International Workshop on Multimedia Signal Processing, Siera, Italy, September 29–October 1 2004.