Instantaneous mental image search
with range queries
on multiple region descriptors

Julien Fauqueur

CUED/F-INFENG/TR 512

January 2005

University of Cambridge
Department of Engineering
Trumpington Street
Cambridge CB2 1PZ
United Kingdom

Email: jf330@eng.cam.ac.uk

Instantaneous mental image search with range queries on multiple region descriptors

Julien Fauqueur

http://www.eng.cam.ac.uk/~jf330/

University of Cambridge
Department of Engineering
Trumpington Street
Cambridge CB2 1PZ

Abstract

The Mental Image Search paradigm allows the user to retrieve images which match the target image s/he has in mind without a starting example. We present a novel approach for this paradigm which enables multiple descriptor range-query, which is necessary to match the more or less precise idea of the user's mental image. In a simple and intuitive way, sophisticated queries can be formulated on the visual appearance of the mental image components. The query interface maps directly to the region multifeature space using a grid file index. Images are retrieved instantaneously on a large photostock database. Various search scenarios show that the combination of this indexing scheme combined with the query interface is an efficient approach for mental image search.

1 Introduction

Searching for images in Digital Libraries is becoming more and more challenging due to the growth of the available content and the increasing number of domains of application in which digital imaging is becoming affordable. The query-by-example was the first CBIR paradigm [6, 10] and, by far, the most common in the state of the art research work. As pointed out by Boujemaa and al. [1], this paradigm "is not able to satisfy the multiple search requirements". In particular, when the user does not have an available example, they propose the "mental image search" paradigm as an alternative. When the user has an idea, or a memory of the image s/he is looking how can a system reach this mental image? They proposed the "boolean composition of region categories" approach [4]. Although our approach shares some similarities with the latter, it differs in most aspects as we will see in section 6. We claim that a system implementing this paradigm should satisfy both following challenging aspects:

- *rich user expression* : in the absence of image example, the user must have the possibility to specify a wide range of visual content with the query interface.
- *simple interaction* : the system must be intuitive, simple and fast to allow the user to perform iterative search if necessary.

Existing systems do not meet both points simultaneously. The popular Query-by-Example paradigm requires simple interaction but provides poor user expression. On the other hand, Query by Sketch approach [8, 3] enables rich user expression but usually requires complex drawing interaction.

To address these aspects we present a simple approach to search images by their regions by specifying ranges of feature attributes. Regions are detected by unsupervised segmentation. Indexing and searching is performed in the quantised multi-feature space where each region is viewed as a multidimensional point. Regions which fall in the same quantised bucket are then similar with respect to all features. Following the basic principle of grid file [7], the constructed index in this space is a multidimensional array which stores the list of regions will fall in each bucket. Thus this index gives direct access to the list of regions which have any given combination of features specified by the user. By construction, this index structure makes range query with respect to all features straightforward, by accessing contiguous buckets in the indexing space. The range query mechanism is necessary to match the *more or less precise idea* of the user's mental image. The associated query interface is designed to directly map the quantised value of each feature. Note that other efficient indexing structures have been used in a query by example context such as R-tree in QBic [6], inverted files [9][4] or multiple inverted array [11].

In section 2 we will explain the indexing approach, including the region segmentation and feature extraction. The range query scheme using the multidimensional index will be explained in section 3. The design of the user interface will be detailed in section 4. We will show the performance of the system in the results section 5 with some practical scenario. Discussion and perspectives will be addressed in section 6 and conclusion in section 7.

2 Building the multi-descriptor index

In this section we present the image indexing process which comprises the following steps: image region segmentation, region feature extraction and quantisation and finally multidimensional index array creation.

2.1 Region segmentation

The goal of image segmentation is to automatically split images of the database into a content-adaptive set of visually salient components which may constitute relevant search keys for the image. We used the unsupervised algorithm proposed by Fauqueur and al. [5] in a content based image retrieval context. It detects coarse regions which are homogeneous with respect to the local distribution of quantised color feature. In our tests we used a 9995 image Corel database from which 50220 regions were automatically detected.

2.2 Partitioning the region feature space

For each segmented region of each image, we extract its visual descriptors. As later explained in section 4, we choose four simple, yet efficient and intuitive, region visual descriptors:

- average color (3-dimensional): average value of the region RGB pixel values
- position (2-dimensional): normalised (x,y) coordinates of the region center mass within the image. The coordinates are normalised with respect to image width and height
- texture (1-dimensional): first RGB color moment (the lower the more uniform the region)
- size (1-dimensional): ratio of number of region pixels by total number of image pixels

All four features are merged into a single 7 (3+2+1+1) dimensional feature vector x which lies in the super feature space $\mathcal{F}$, defined as the cartesian product of all the region feature spaces. A region is then seen as a 7 dimensional point x in $\mathcal{F}$. Let D be the dimension of this super space and let us write x as $(x_1, \ldots, x_D)$. During the indexing step the D dimensions are processed all together regardless the feature they are part of. We assume all x_i have 0 and b_i as theoretical bounds, i.e. $\forall i, x_i \in [0; b_i]$.

A simple uniform quantisation of all D dimensions is performed. Each dimension i is reduced to a number K_i of values. The choice of the number K_i will be discussed later. If we denote as $\mathcal{F}'$ the quantised set, we have $\mathcal{F} = [0; b_1] \times \ldots \times [0; b_D] \subset \mathbb{R}^D$ and $\mathcal{F}' = \{0, \ldots, K_1 - 1\} \times \ldots \times \{0, \ldots, K_D - 1\} \subset \mathbb{N}^D$. The quantisation process is viewed as an application from $\mathcal{F}$ to $\mathcal{F}'$:

$Q : \mathcal{F} \mapsto \mathcal{F}'$

$Q : (x_1, \ldots, x_D) \mapsto (q(x_1), \ldots, q(x_D))$, where $q(x_i) = \lfloor x_i.(K_i - 1)/b_i \rfloor$ and where $\lfloor x \rfloor$ rounds x downwards to the nearest integer. Quantised values $q(x_i)$ are in $\{0, \ldots, K_i - 1\}$.

After the quantisation process, each region is associated with a D dimensional vector of quantised feature values in the space $\mathcal{F}'$. In the D-dimensional space, quantisation has the effect of grouping regions with similar superfeatures into the same hypercubes, referred to as *buckets*. Fine quantisation (i.e. high values of K_i) results in a fine granularity and only regions with highly similar features are grouped together. Conversely, coarse quantisation groups together regions which are moderately as well as highly similar. As pointed out in [7], the resulting partition becomes highly sparse. The total number of buckets is $\prod_{i=1}^{D} K_i$ which grows exponentially with the number of dimensions (seven in our tests). Also the ratio of the number of empty buckets by the total number of buckets grows asymptotically. In [7] they recommend the use of grid files for up to 10 dimensions which is suitable for our application. In figure 1, we also represented the ratios of empty and non-empty buckets as the number of quantisation levels grows for the 50220 segmented regions. The graphs are plotted for 7 dimensions and the same number of quantisation levels for all dimensions. We also observe that, as the number of quantisation levels increases, the ratio of empty buckets grows asymptotically to 1 and the ratio of non-empty ones decreases asymptotically to 0. This figure also shows the decline of the average bucket cardinality (the number of regions they contain). If the number of quantisation levels or the dimension is too high, buckets will become

very small and tend to contain a single region. For the sake of query and retrieval effectiveness, the design of the query interface (see section 4) suggested the use of 4 and 5 quantisation levels. This quantisation granularity applied to the 50220 region database yields a total of 40000 buckets of which 90% are empty. The average cardinality of the non-empty buckets is 12.98 regions.

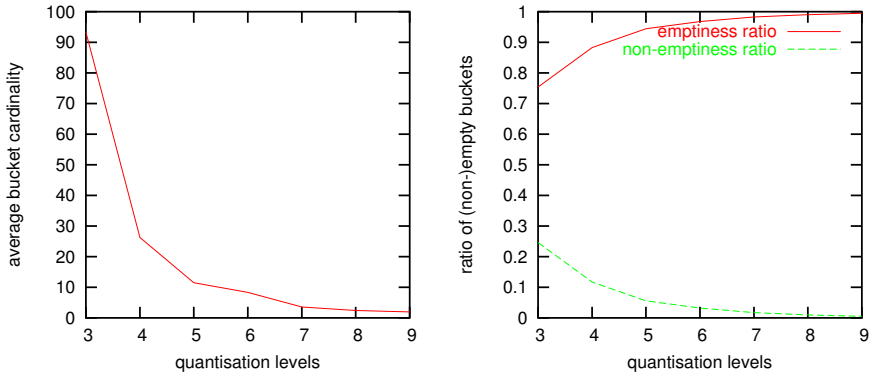

Figure 1: Bucket cardinalities with respect to the number of quantisation levels: average cardinality of non-empty buckets (left) and ratio of the number of empty and non-empty buckets by the total number of buckets (right). The fast asymptotical decline of both average bucket cardinality and non-empty ratio motivates a rather small number of quantisation levels.

2.3 Associating regions to buckets with the index array

Given the previously detailed procedure, all region features of the database are quantised. Each region hence falls into a unique bucket in the D-dimensional quantised space. As we will see in section 4, the user input to the system will consist of a list of quantised values for each feature. The retrieval scheme will consist in finding regions which fall into the buckets corresponding to that values. The indexing structure we are presenting now aims at giving the list of regions within a bucket given any D dimensional vector of quantised values. It follows the basic principle of grid files [7]. This indexing structure is a multidimensional array A such that for all $(a_1, \ldots, a_D) \in \mathcal{F}'$, $A(a_1, \ldots, a_D)$ gives the list of regions which fall into bucket $(a_1, \ldots, a_D)$. Figure 2 illustrates an example of a three dimensional quantised space in which $A(a_1, a_2, a_3)$ points to a bucket containing three regions. The algorithm to construct array A is the following:

- initialise A: $\forall (a_1, \ldots, a_D) \in \mathcal{F}', A(a_1, \ldots, a_D) = \emptyset$
- $\forall$ region R of quantised vector $x_R = (a_1, \ldots, a_D)$, do: $A(a_1, \ldots, a_D) = A(a_1, \ldots, a_D) \cup \{R\}$

At the end of this process, the array A will generally be very sparse, i.e. most of the buckets $A(a_1, \ldots, a_D)$ will remain empty while the others will contain a variable number of regions. The content of the array for a given database of regions is stored in a file as shown in figure 3. Since we only store the non empty buckets, the size of the index file will virtually not be affected by the array sparsity. Each row of the index file contains the bucket index, its cardinality and the list of identifiers of the regions it contains. In a separate file, we also store the region filenames (whose

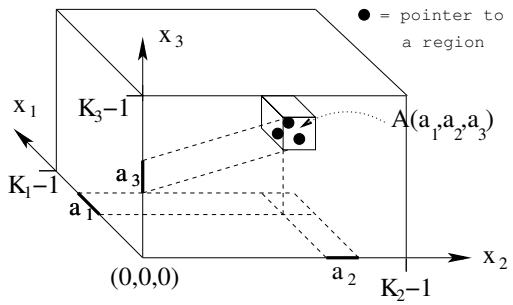

Figure 2: Index array A directly points to the regions with quantised feature values (a_1, a_2, a_3). $D = 3$ in this example.

syntax also provides the image filenames) and their identifiers. The size of this separate file does not depend on the number of feature dimensions of quantisation levels.

index	bucket cardinalities	region identifiers
⋮		
2 1 3 0 1 0 1	1	21930
3 1 3 0 1 0 1	6	8676 19283 26538 35310 39597 39770
⋮		

Figure 3: The index file stores one row per non-empty bucket. Each row is composed of the bucket index (first D columns), its cardinality and the identifiers of regions contained in the bucket.

Using this index array A, we now see that retrieving regions which have a particular set of attributes $(a_1, \ldots, a_D)$ is as simple as accessing the bucket $A(a_1, \ldots, a_D)$ and does not involve any loop or particular computation. By construction, this array A gives the content of any bucket of the quantised whether it be empty or not. We see that the advantage of this representation is to keep the index file size low while allowing us to access directly the content of any bucket, although the total number of buckets can be very large.

3 Range query with the multi-descriptor index array

In the previous section we presented the construction of the index array which gives us direct access to the list of regions which satisfy a given set of attributes $(a_1, \ldots, a_D)$. In a query scenario, this would mean, for instance, that we can directly retrieve regions which have a given color, texture, position and size. For instance to search sky areas, the user may want to specify big, blue and top/center regions. But what if the user thinks a feature is not relevant for a particular query? For instance, to search skin parts, the color may be relevant but not the size and position. So the user may also want to be able to query from a partially specified set of attributes, to express that an attribute is somewhat relevant ("somewhat centered within the image") or completely irrelevant ("any position within the image"). This means searching the partial range or the full range of any attribute. The first case where all attributes are explicitly specified is referred to as

a *fully specified query* and the second as a *partially specified query*. The first case will be naturally handled as a particular case of the second. The previous figure 2 shows that a fully specified query (the query being the triple (a_1, a_2, a_3)) is solved by a single bucket access with the index array A. In figure 4 we show the set of buckets which corresponds to a range query. In a general case, we

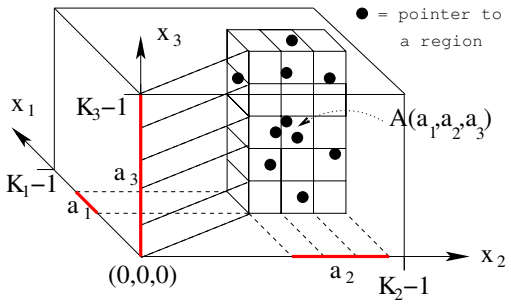

Figure 4: Range query in a 3D feature space around bucket $A(a_1, a_2, a_3)$. It is decomposed into a point query around a_1, a 1-nearest-neighbor around a_2, and a full range query along x_3 axis.

write a user query $\mathcal{Q}$ as a list of subsets of values for all dimensions:

$$\mathcal{Q} = \{S_1, \ldots, S_D\} \tag{1}$$

where $\forall i = 0, \ldots, D$, S_i is any subset whose elements belong to $\{0, \ldots, K_i - 1\}$. The S_i correspond to the quantised values for each feature dimension i. For example, the range query illustrated in figure 4 is written as: $\mathcal{Q} = \{\{a_1\}, \{a_2 - 1, a_2, a_2 + 1\}, \{0, \ldots, K_3 - 1\}\}$. The result set of regions which match the query is then:

$$\begin{aligned}\mathcal{R} =& A(a_1, a_2 - 1, 0) \cup A(a_1, a_2, 0) \cup A(a_1, a_2 + 1, 0) \cup \\ & \vdots \\ & A(a_1, a_2 - 1, K_3 - 1) \cup A(a_1, a_2, K_3 - 1) \cup A(a_1, a_2 + 1, K_3 - 1)\end{aligned}$$

In general, solving a query $\mathcal{Q}$ (formula (1)) comes down to determining the unions:

$$\begin{aligned}\mathcal{R} &= \bigcup_{a_1 \in S_1} \cdots \bigcup_{a_D \in S_D} A(a_1, \ldots, a_D) \\ &= \bigcup_{(a_1, \ldots, a_D) \in \mathcal{Q}} A(a_1, \ldots, a_D)\end{aligned} \tag{2}$$

where subsets S_i are provided by the user through the query interface and where A is the index array. Note that the order of the union operators (i.e. the order of dimensions) can be swapped without affecting the result $\mathcal{R}$.

4 User query interaction

In this section we present the query and results interface and detail the user interaction methodology. The graphic user interface is designed in order to let the user choose the relevant features

which best describe the region in the target image. In the particular context of mental image search, the query interface is an essential aspect of the system, as it should easily let the user specify the visual content of the target image without prior example. The key aspect of its design is to make the query formulation *intuitive* because the user is not expected to be an image processing expert and *quick* so that iterative search can be easily performed, if necessary.

4.1 Feature palettes for the query interface

The query interface must assist the user to formulate the query $\mathcal{Q} = \{S_1, \ldots, S_D\}$, i.e. to select the list of quantised values for each feature. A screenshot of the query interface is shown in figure 5. For each feature we build a dedicated palette, referred to as *feature palette*, which consists of a list of checkboxes corresponding to each of the feature quantised values. For scalar features, such as region size and texture, corresponding palettes are a simple linear arrangement of five boxes corresponding to the five possible quantisation values. Above the checkboxes, small icons provide a graphic illustration of the quantisation levels: for region texture, a grey uniform icon depicts the lowest value and a highly textured icon depicts the highest one. For region size, each quantisation value is illustrated by the size of a white square on a black background icon. The region position comprises a total of 25 possible values (5 per spatial dimension). The layout of the 25 checkboxes in a square corresponds to the image area.

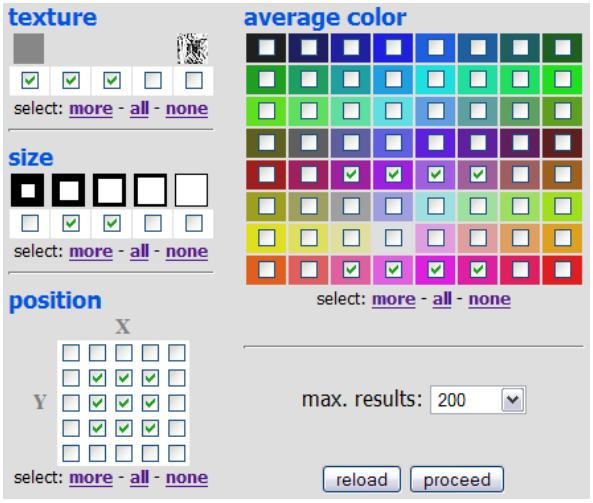

Figure 5: Query interface and feature palettes. All possible quantised values of each feature are displayed and can be (de-)selected individually or by group. Each palette has its "more/all/none" buttons for the range query mechanism (see section 4.2).

For features of dimension 3 and above, such as average color, the problem is to represent the possible quantised values within the bidimensional graphic user interface. The RGB average color feature has $4^3 = 64$ possible colors which we display in lexicographic order: increasing R, then increasing G then increasing B. The overall smooth transition is achieved by alternating increasing and decreasing values for both G and B. Other color picker tools can be adopted such as those reviewed in [2].

The number of quantisation levels K_i for all features was set to 4 for color and 5 for the others, after testing various possibilities. The choice of number of levels have two opposite effects on the query interface and retrieval precision. A too high number of levels results in a very high number of quantisation values to choose from in the query interface. The user may hesitate to choose between two values if they are too similar. On the other hand, a too low number of levels means that the quantisation is coarse and regions with dissimilar visual appearance fall into the same bucket. This would result in poor retrieval precision. In the particular case of colors, more than 4 quantisation levels would produce too many redundant colors in the palette and less than 4 would make that regions with different colors would be grouped together. 4 was then found to be a good choice for color. Besides we found that an odd number of quantisation levels was important for 1- and 2- dimensional features in order to have a "middle" value, particularly for the position.

We see here the importance of using simple visual descriptors so that their respective feature palettes are intuitive. More sophisticated descriptors, such as color histograms or advanced texture descriptors, may be inadequate for a mental image search query interface. Indeed building a feature palette for a high dimensional descriptor is complex and is likely to result in a non-intuitive interface. For example in QBic [6] an interface was designed to let the user select a combination of a few color percentages to specify a color distribution in the query. This assumes the user is familiar with the concept of color distributions and knows how to specify a distribution which is relevant for the target image. In VisualSeek [8] to specify a texture as a query, a library of different texture patches was presented as a list to browse. Since texture in a generic database can have an highly different aspects, browsing to find the right query texture is very tedious. Unless very efficient feature palettes for high dimensional descriptors can be designed, we think that combining different simple features is a key factor to have an efficient query user interface while enabling specific visual queries.

4.2 Formulation of multi-descriptor range query

The range query mechanism is an essential aspect of mental image search to express more or less vague queries. For a given target region the user may want to select more or less values of a feature to broaden or narrow the search. To search in the upper part of an image we may want to select the top first 10 position boxes, whereas for a search at the specific center position we may want to check the single center position box. As shown in figure 5, in each feature palette, the three "more-all-none" buttons assist the user in the selection of corresponding value checkboxes:

- "more" implements an "automatic range query selection" by expanding the query by 1 nearest neighbor in the quantised feature space around the currently selected value. It automatically checks the boxes. The automatic selection is achieved instantaneously since it does not involve access to the array but simply relies on an adequate naming of checkboxes.
- "all": checks all the values of the corresponding feature. It is used to perform a full range query on the feature, if a feature is judged irrelevant for a query.
- "none": deselects all values.

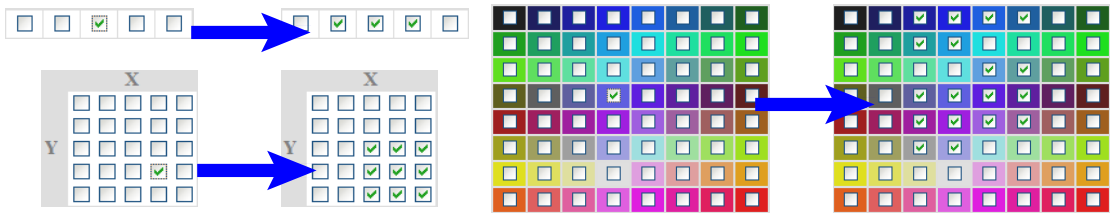

Figure 6: Automatic range query selection with the "more" button for all the features. In each case, one box is checked by the user and further clicking the "more" button makes the automatic checking of the contiguous neighbor in the feature space. Note that color neighbors in the 3D space are not always contiguous in 2D.

The value automatically checked with the "more" button ensures that the closest values in the feature spaced are taken into account in the range query. In practice this mechanism was found to be very helpful.

4.3 Results interface

Once all the feature values have been selected, pressing the "proceed" button in the query interface will retrieve all images which have a region matching the query $\mathcal{Q}$. From the set of checked boxes from the query interface, we have the list of range values for each feature dimension which were earlier denoted as $S_1, \ldots, S_D$. We now want to display the set $\mathcal{R}$ of regions defined in equation (2). The process of determining and displaying the relevant images is the following:

$\forall a_1 \in S_1$, ..., $\forall a_D \in S_D$, $\forall$ region $R \in A(a_1, \ldots, a_D)$, display the image that contains R

Note that $A(a_1, \ldots, a_D)$ stores a list of regions and that we display the images that contain these regions. The association between regions and images is provided by the way regions have been named. Between two series of images from a same bucket, a blank space is inserted to distinguish the different groups of images. Only the first M images are displayed, where M is the maximum number of results value set in the query interface (see the "max. results" box in figure 5).

5 Results and Scenarios

We used the Corel database of 9995 segmented images provided by IMEDIA group at INRIA[1].

As samples show in figure 7, the content of this database is heterogeneous : landscapes, portraits, objects, flowers, cars, animals, kitchens, food, etc. A total of 50220 regions are detected (average of 5 regions/image). The code for the server was written in CGI-C^{++}, and the client interface in HTML and javascript. The system ran on a 700MHz PC.

[1] http://www-rocq.inria.fr/imedia/

Figure 7: Overview of Corel database.

5.1 Retrieval performance

In the context of the mental image search paradigm, the main goal is to retrieve a relevant set of images given a more or less precise idea or memory of an image the user has in mind. A finer search can further be performed using a classic query by example engine on our result set of images using more sophisticated descriptors. We present the results of five retrieval scenarios of scenes. They show how various types of scenes can be easily retrieved either by searching the area of interest itself as well as the background. For each scene we describe the region visual attributes we thought were relevant for the search:

target scene	search region	average color	size	position	texture
"grass"	grass	green	any	bottom	low
"sunny skies"	blue sky	deep blue	medium	top-center	low/medium
sea/swimming pool	sea or pool patch	various shades of blue	big	bottom	low-medium
objects on white backgrounds	white background	white	big	center	low
"fireworks"	black background	black	big	center	low

All these visual attributes were translated into queries in our feature palettes. In the appendix section, in figures 9 to 13, we present the screenshots of the actual queries and the retrieved images (in most cases only top results are shown). The "grass" query was the easiest query since it seems the presence of a green region in the bottom of an image is very typical of grass. The "sea/swimming pool query" required a few query/retrieval iterations to adjust the colors to a relevant range of blue. Two swimming-pool scenes and a few sea scenes are retrieved. There seems to be two reasons for this low number of results: these scenes are underrepresented in the database and their visual appearance is very variable (especially for the sea). The "object on white background" query retrieves images which have a semantic consistency of studio photographs of objects or models on white backgrounds. To retrieve fireworks, a search on big black centered regions helped find night scenes which happened to contain fireworks. The last two queries ("fireworks" and "studio

images") show that searching by the background can be efficient when searching the object of interest itself is difficult.

All five presented search results contain many relevant images. They are likely to be similar or correspond to the user mental image. For more difficult queries, if very few retrieved images are relevant, a relevant image may be used as an example to perform a classic query by example search which involves finer descriptors. So our mental image search engine can be used as such or in combination with a classic query by example engine for query refinement.

5.2 Instantaneous retrieval

Over 23 queries, we determined the retrieval time depending on the number of retrieved images on the 700MHz PC. This time measure grows with the number of retrieved images. The highest time was 0.48 second when a full range query on all 4 features was performed. All the 50220 regions were returned. A query is considered as intantaneous when it is processed in 0.1 second at most [7]. The queries which took longer than 0.1 second were those returning 15000 regions or more. In practice, we only display the first 200 retrieved images which takes 0.01 second at most. These retrieval times are very low given that range queries on multiple region features are performed.

6 Discussion and perspectives

6.1 Distribution of quantised features

By using the system we observed that depending on the selected quantised values we may have very different numbers of results, from none to sometimes thousands. These extreme cases are unsatisfactory from the user point of view. To understand the variable number of results we studied the distribution of regions in the buckets.

Figure 8 illustrates the distribution of the number of regions which fall into the same bucket for each feature. For each of the four features and each quantised value of the feature, we plotted the number of regions which have these feature values and any value for the other feature (it is performed by range query on the other three features). The color distribution is shown as the 64 colors along with their corresponding bucket cardinalities sorted by increasing cardinality (top left figure). We observed the following facts for the different features:

- color (top left figure) : as in [4], we observe that black bucket (last bucket in bottom row) has by far the highest cardinality. Globally, the highest cardinalities are observed in low color saturation buckets (black, grey, white) and the lowest cardinalities in high saturation buckets (in particular bright green, bright blue, bright magenta buckets). Buckets with average cardinalities (buckets in middle rows) tend to embed most of the natural colors (i.e. colors of medium saturation).

- size and texture (right figures): their distribution are strongly decreasing. Most of the regions lie in the first bins, i.e. most of the regions are somehow small and with low texture.
- position (bottom left figure): the fact this feature distribution is more "uniform" than the others is quite consistent with the likeliness that a region can a priori be located anywhere in an image. The central peak (in index 2,2) is explained by the fact that the bigger a region is and the closer to the image center its barycentre is.

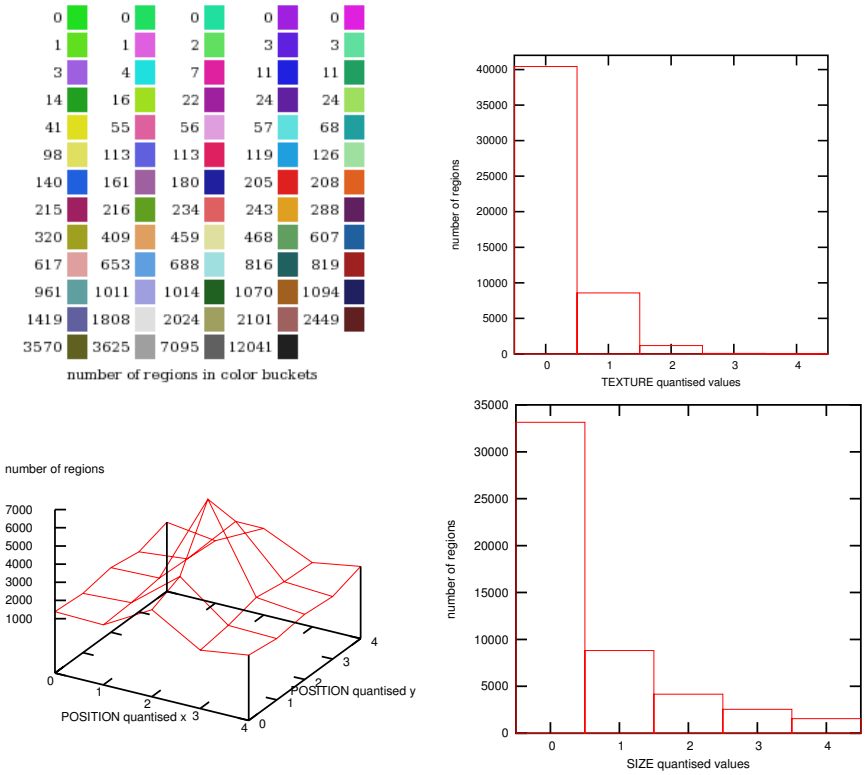

Figure 8: Distribution of the number of regions with respect to the four feature quantised values.

The unevenness of bucket cardinalities is due to the nature of features themselves, to the uniform partitionning of the feature space and also to the specificity of the database content. Investigating some transformations on the features may make the distributions somehow more uniform by making them perceptually uniform. In that regard, we plan to use the perceptually uniform color spaces Luv of Lab instead of RGB. For the other features, a perceptual partitionning to the various areas of the feature space may also be investigated.

6.2 Related work

The "boolean composition of region categories" approach [4] is another implementation of the mental image search paradigm. Categories of regions are formed by unsupervised clustering of their color descriptors. Indexing with inverted files on the region category labels allows the user to formulate boolean queries on the types of regions which should be present or absent in the mental

image. The query interface consists of a visual summary of available regions in the database. Range queries in the region color feature space are achieved by letting the user specify a numerical value for the range radius. This approach and ours both implement the mental image search paradigm and both rely on efficient indexing structures inspired from the information retrieval framework. But they differ in many aspects.

In their approach, performing a feature clustering provides an intuitive visual summary with respect to the region features. This may be an advantage to have an overview of the database but it makes range queries and the integration of multiple features harder to implement. In our approach the region grouping is performed by uniform quantisation in the region feature space. It is faster and simpler to achieve than clustering and handles multiple features as well as range queries in a simple and intuitive way. Our query interface does not depend on the database content. Having a constant query interface is an advantage make the query easier for the user when searching in different databases but, on the other hand, it does not help perceiving the overall database content. A combination of constant and adaptive interface may be a way to have both advantages. We also plan to allow for boolean queries with multiple regions as they proved to capture efficiently the scene semantics in the "boolean composition of region categories" approach. They can be very easily implemented with our index array.

6.3 Perspectives

In the future, we plan to study the three following aspects: (i) investigate the perceptual uniformity of region features to improve the feature partition, (ii) allow for boolean composition on multiple regions and (iii) test this approach on bigger databases.

7 Conclusion

We presented a new approach for mental image search which enables rich user expression and simple interaction. Indeed, the mental image can be retrieved by specifying multiple visual descriptors of a region composing the image. Range query can be easily performed to match the more or less precise idea of the mental image apperance. Furthermore, user query interaction is reduced to a few clicks and search is performed instantaneously on a large database. This was made possible by the joint use of a multi-descriptor indexing structure based on grid file and of a simple user interface which directly maps the index. Retrieval examples illustrated the ability of this approach to retrieve scenes corresponding to various types of mental images.

Acknowledgements

This work has been carried out with the support of the UK Data and Information Fusion Defence Technology Centre. The author would like to thank Nick Kingsbury, Marta Wilczkowiak and Ryan

Anderson for their fruitful feedback and IMEDIA group for providing the segmented database.

References

[1] N. Boujemaa, J. Fauqueur, and V. Gouet. *What's beyond query by example? book chapter from* Trends and Advances in Content-Based Image and Video Retrieval*, L. Shapiro, H.P. Kriegel, R. Veltkamp (ed.).* LNCS, Springer Verlag, 2004.

[2] E. L. Broek, P. M. F. Kisters, and L. G. Vuurpijl. Design guidelines for a content-based image retrieval color-selection interface. *ACM Conference on Dutch directions in HCI*, 2004.

[3] A. DelBimbo and P. Pala. Visual image retrieval by elastic matching of user sketches. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2), february 1997.

[4] J. Fauqueur and N. Boujemaa. Mental image search by boolean composition of region categories. *to appear in Multimedia Tools and Applications*, 2004.

[5] J. Fauqueur and N. Boujemaa. Region-based image retrieval: Fast coarse segmentation and fine color description. *Journal of Visual Languages and Computing (JVLC), special issue on Visual Information Systems*, 15(1):69–95, 2004.

[6] W. Niblack, R. Barber, W. Equitz, M. Flickner, and al. The QBic project: querying images by content using color, texture, and shape. *Proc. SPIE (Storage and Retrieval for Image and Video Databases)*, 1908:173–187, 1993.

[7] J. Nievergelt, H. Hinterberger, and K. C. Sevcik. The grid file: An adaptable, symmetric multikey file structure. *ACM Transactions on Database Systems (TODS)*, 9(1):38–71, 1984.

[8] J. R. Smith and S. F. Chang. Visualseek: A fully automated content-based image query system. *ACM Multimedia Conference, Boston, MA, USA*, pages 87–98, 1996.

[9] D. Squire, W. Muller, H. Muller, and J. Raki. Content-based query of image databases, inspirations from text retrieval: inverted files, frequency-based weights and relevance feedback. *11th Scandinavian Conference on Image Analysis (SCIA) Kangerlussuaq, Greenland*, 1999.

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

[11] N. Taniguchi, H. Akama, and M. Yamamuro. Multiple inverted array index structure for asymmetric similarity measure. *Proceedings of Challenge of Image Retrieval*, 1998.

Appendix: results of mental image search scenarios

In this section we show five examples of mental image search scenarios. Each figure illustrates the quantised features used for the query and the top-ranked retrieved images. For all figures but figure 11, only first images are shown for space reasons.

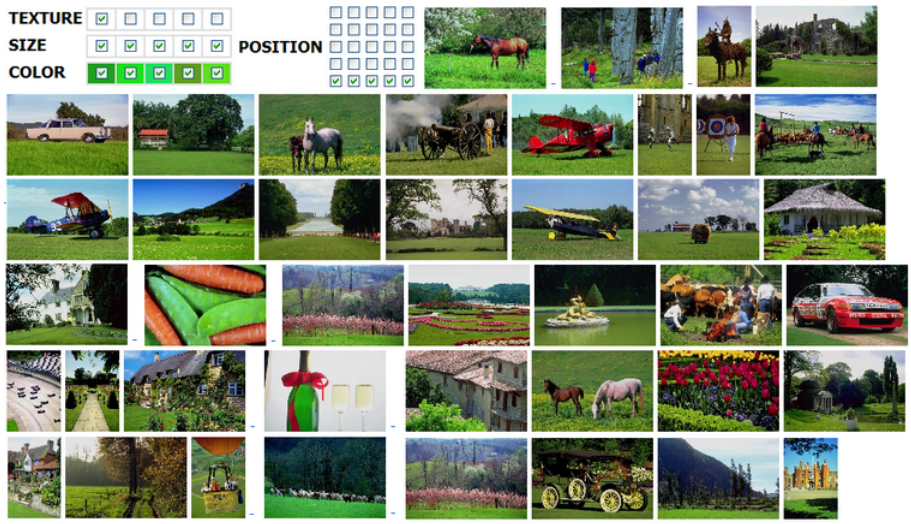

Figure 9: "grass" query.

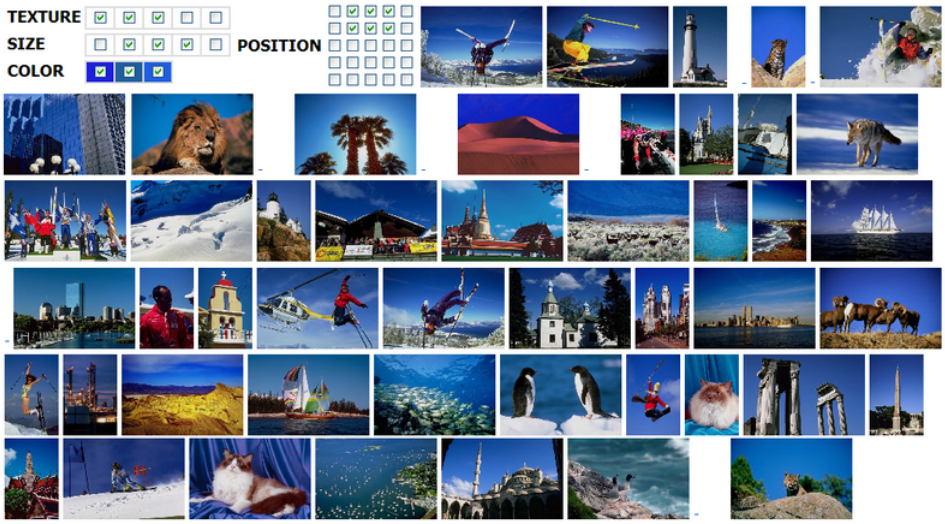

Figure 10: "sunny skies" query.

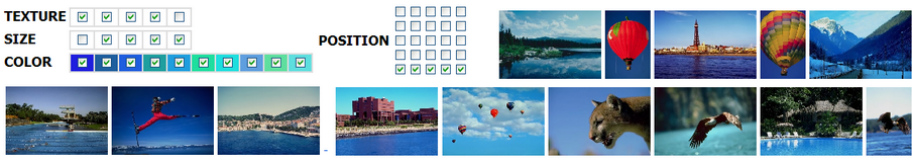

Figure 11: "sea/swimming-pool" query.

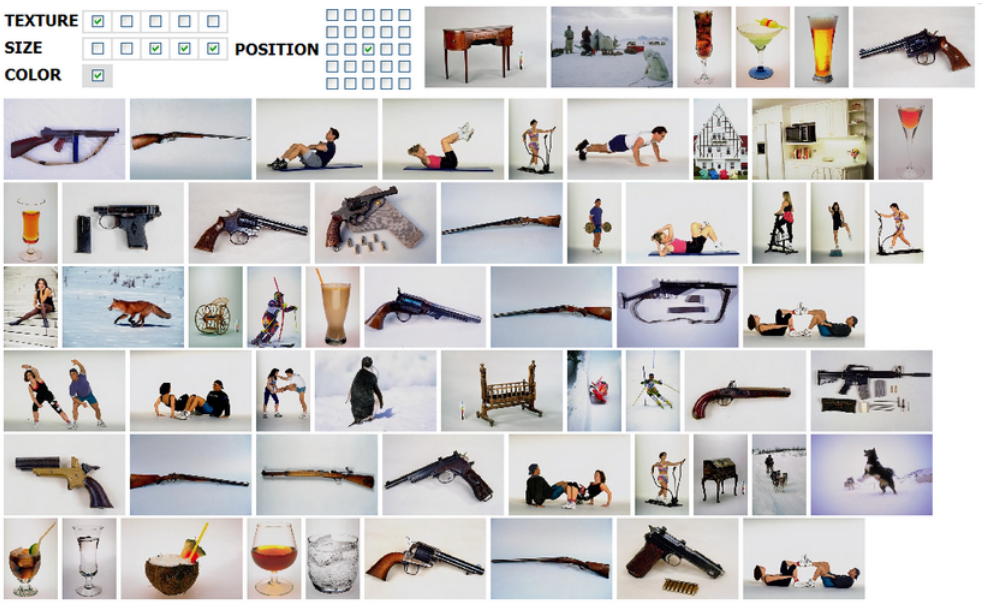

Figure 12: "object on white background" query.

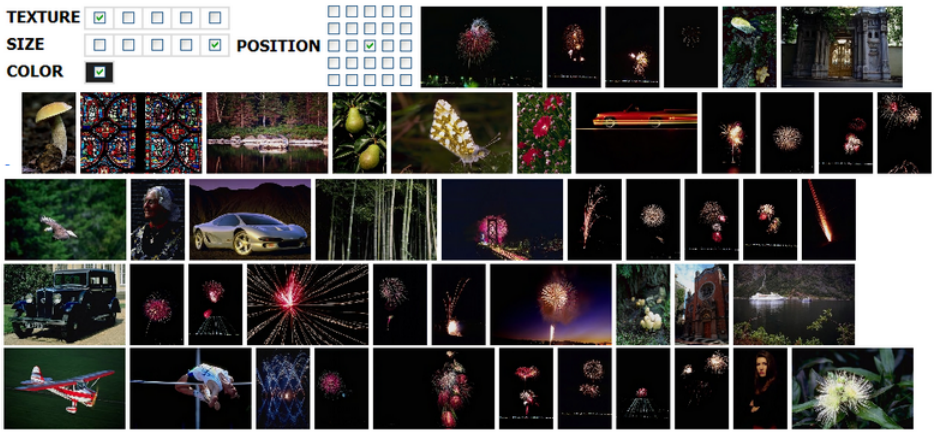

Figure 13: "fireworks" query.