Multimedia Systems (2005) 10(6): 529–543
DOI 10.1007/s00530-005-0180-9

REGULAR PAPER

Ruofei Zhang · Zhongfei (Mark) Zhang

FAST: Toward more effective and efficient image retrieval

Published online: 20 April 2005

Abstract This paper focuses on developing a Fast And Semantics-Tailored (FAST) image retrieval methodology. Specifically, the contributions of FAST methodology to the CBIR literature include: (1) development of a new indexing method based on fuzzy logic to incorporate color, texture, and shape information into a region-based approach to improving the retrieval effectiveness and robustness; (2) development of a new hierarchical indexing structure and the corresponding hierarchical, elimination-based A* retrieval (HEAR) algorithm to significantly improve the retrieval efficiency without sacrificing the retrieval effectiveness; it is shown that HEAR is guaranteed to deliver a logarithm search in the average case; (3) employment of user relevance feedback to tailor the effective retrieval to each user's individualized query preference through the novel indexing tree pruning (ITP) and adaptive region weight updating (ARWU) algorithms. Theoretical analysis and experimental evaluations show that FAST methodology holds great promise in delivering fast and semantics-tailored image retrieval in CBIR.

Keywords Content-based image retrieval · Region-based features · Hierarchical indexing structure · Indexing tree pruning · Relevance feedback

1 Introduction

This work addresses the topic of general-purpose content-based image retrieval (CBIR). CBIR has received intensive attention in the literature since this area was started a few years ago, and consequently a broad range of techniques [26] is proposed.

The majority of the early research focused on global features of imagery. The most fundamental and popularly used feature is the color histogram and its variants, which was used in the classic systems such as IBM QBIC [13] and Berkeley Chabot [29]. Since color histograms do not carry spatial information, which is considered to be related to the semantics of image content, efforts have been reported in the literature to incorporate the spatial information into the histograms. Pass and Zabih [31] proposed the color coherence vector (CCV) to address this issue. Huang et al. [20] went further, using the color correlograms to integrate color and spatial information.

Recently region-based approaches have been shown to be more effective. A region-based retrieval system segments images into regions and retrieves images based on the similarities derived from the regions. Berkeley Blobworld [5] and UCSB NeTra [25] compared images based on individual regions. To query an image, the user was required to select regions and the corresponding features for the similarity computation. Wang et al. [44] proposed an integrated region matching scheme called IRM that allowed for matching a region in one image to several regions in another image. As a result, the similarity between two images was defined as the weighted sum of distances, in a feature space, between all regions from different images. Later, Chen and Wang [7] proposed an improved approach called UFM based on applying coarse fuzzy logic to different region features to improve the retrieval effectiveness of IRM. Jing et al. [12] presented a region-based, modified inverted file structure analogous to that in text retrieval to index an image database; each entry of the file corresponded to a cluster (called codeword) in the region space. While this method is reported to be effective, the selection of the size of the code book is subjective in nature, and the effectiveness is sensitive to this selection. More recently, Zhang et al. [47] proposed a hidden semantic concept discovery technique based on the region representation of an image database by developing a generative model. Image retrieval is based on the similarity of discovered concepts, which is shown to be more reliable and effective than that based on original region features.

To narrow the semantic gap in image retrieval, several recent efforts in CBIR, such as [15] and [8], performed the

R. Zhang · Z. M. Zhang (✉)
Department of Computer Science, Sate University of New York at Binghamton, Binghamton, NY 13902, USA
E-mail: rzhang@cs.binghamton.edu, zhongfei@cs.binghamton.edu

image retrieval based not only on the content but also on user preference profiles. Machine learning techniques such as support vector machine (SVM) [41], Bayesian network [36], and decision tree [49] were applied to learn a user's query intention through leveraging preference profiles or relevance feedback. One drawback of such approaches is that they typically work well only for one specific domain, e.g., an art image database or a medical image database. It has been shown that for a general domain the retrieval accuracy of these approaches is weak [26]. In addition, these approaches are limited by the availability of user preference profiles and the generalization limitation of machine learning techniques they apply.

While the majority of the literature in CBIR focuses on the indexing and retrieval effectiveness issue, much less attention has been paid to the indexing and retrieval efficiency issue. Historically, CBIR research was motivated initially by a desire to demonstrate that indexing directly in the image domain could deliver better retrieval effectiveness than indexing through collateral information such as keywords, and this perception has been perpetuated over the years; retrieval efficiency, on the other hand, is not considered to be the focus of research in the CBIR community, as the general perception is that this issue may be resolved by directly making use of the existing indexing methods developed from the spatial data structure research.

Several spatial data structures have been proposed in the literature. Of these data structures, some are based on the ideas of B^+- and B-trees [11], which initially are for organizing 1D feature vectors or single valued keys of stored items, such as multidimensional B^+-tree [9], while others perform feature searching and updating by "ordering" the multidimensional features either based on feature space partition and filtering, such as K-D tree [2], R-tree [34, 17] and its variants, R^*-tree [14], R^+-tree [38], and TV-tree [23], or by similarity measuring [33]. Another data organization method is the grid file [28], by which an n-dimensional space is divided into equal-sized hypercubes with each hypercube containing zero or more feature vectors. In grid file search, the search scope is reduced from the whole feature space to a hypercube (grid) to facilitate data insertion. However, this classic method is not suitable for very high-dimensional vectors, which are common in multimedia processing.

Several problems arise when applying these generic data structures to CBIR directly. First, technically many CBIR algorithms involve complicated distributions in a high-dimensional feature space, and it is difficult and inflexible to directly "order" features in such a high-dimensional space using the existing spatial data structures. Second, while theoretically any CBIR methods can use the existing spatial data structures to address the retrieval efficiency, practically speaking this is not the case because when the dimensionality becomes very high (which is true for almost all the CBIR methods), the overhead for online bookkeeping becomes so demanding [37] that the overall savings in efficiency become questionable. Some data structures, such as SS-tree [45] and TV-tree [23], attempt to address this problem, but their overall performances are limited due to the assumptions they are subject to [24]. Third, the existing spatial data structures do not work well for region-based image indexing approaches due to the fact that the mapping between the region space and the image space is typically nonmetric [7, 12, 44]. Consequently, several efforts in the literature have attempted to address the efficiency issue directly in designing specific CBIR algorithms. For example, Berman and Shapiro [3] used a set of keys along with the triangle inequality in image databases for fast search.

Since the evaluation of CBIR retrieval is typically subjective, in recent years methods incorporating user relevance feedback have started to show promise in resolving this issue. Two directions of research are observed in incorporating user relevance feedback in CBIR: (1) developing a weighting scheme to explicitly "guide" the retrieval [32] and (2) applying machine learning techniques such as Bayesian net and SVM to reduce the problem to a standard reasoning and classification problem [39, 40, 48].

Based on an extensive literature review, we have identified three problems in the current status of CBIR research: (1) indexing effectiveness still needs to be improved, (2) retrieval efficiency needs to be addressed directly in the indexing method, and (3) retrieval subjectivity needs to be further addressed to better deliver a semantic retrieval. This work is motivated by a desire to address these three issues simultaneously. The ultimate goal of this project is to design a CBIR methodology that can deliver fast and semantics-tailored image retrieval capability. By fast, we mean that the efficiency issue is well addressed; by semantics-tailored, we mean that the user query preference is inferred online to allow an individualized retrieval to better address the retrieval effectiveness and user preference issues. Consequently, we call the methodology FAST. The contributions of FAST are reflected in these three aspects. An overview of the FAST architecture is shown in Fig. 1

One system related to FAST is FourEyes developed by Minka and Picard [27]. FourEyes is one of the first systems employing machine learning techniques with region-based image representation. It contains three stages: grouping generation, grouping weighting, and grouping collection. Although both FourEyes and FAST are region based, it is noticeable that there are four major differences between them. First, the approaches employed are different. FourEyes selects groupings of the data through learning best feature models from a "society of models" based on users' interactions. FAST is not designed to select features; instead, FAST integrates several features to improve the robustness of feature representation and to facilitate the subjective retrieval by using fuzzified features and a systematic classification paradigm for learning user retrieval preferences. Second, the self-organization map (SOM)-based weighting algorithm used in the grouping weighting of FourEyes is computation intensive, whereas in FAST the adaptive region weight updating algorithm is more efficient. Third, with a specifically developed indexing structure and

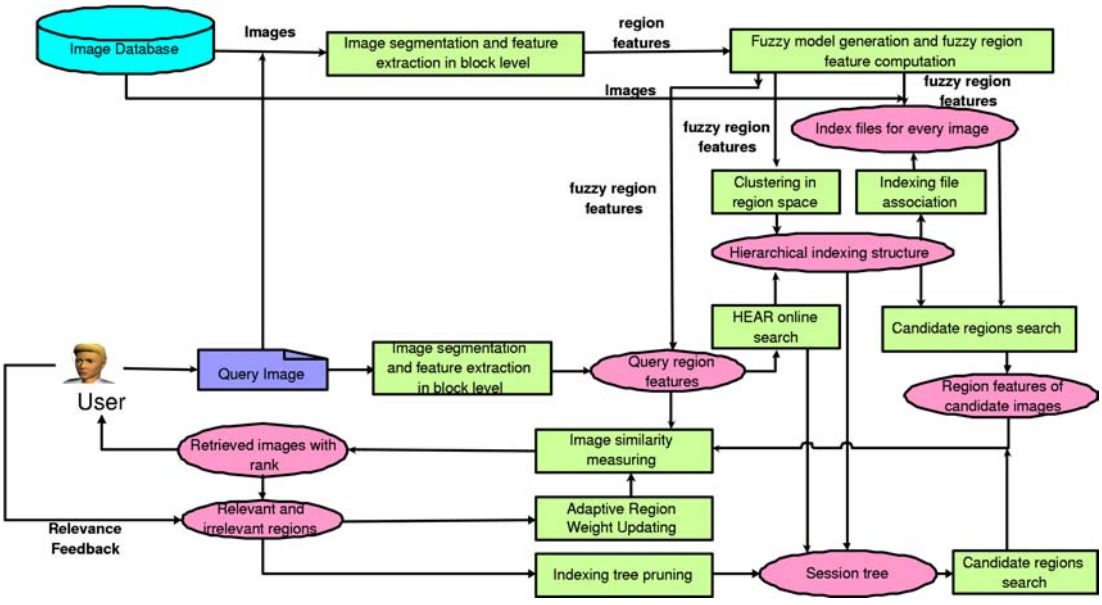

Fig. 1 Overview of the FAST architecture

a corresponding optimized search algorithm, FAST has the benefit of overall lower complexity and higher scalability than FourEyes. Fourth, the objectives of FourEyes and FAST are different. FourEyes is a semiautomated region annotation tool, while FAST aims to provide a comprehensive framework for efficient and effective image retrieval.

The paper is organized as follows. Section 2 describes the fuzzified feature representation and indexing scheme. The region matching and similarity computation metrics are also discussed in this section. The proposed hierarchical indexing structure and the hierarchical, elimination-based A* retrieval (HEAR) online search algorithm are presented in Sect. 3. Section 4 describes the developed indexing tree pruning (ITP) and the adaptive region weight updating (ARWU) algorithms to capture users' retrieval subjectivity. Section 5 presents the empirical evaluations of a FAST prototype. The paper is concluded in Sect. 6.

2 Fuzzified feature representation and indexing scheme

We propose an efficient, clustering-based, fuzzified feature representation approach to addressing the general-purpose CBIR. In this approach we integrate a clustering-based segmentation with fuzzy representation of color histogram, texture, and shape to index image databases.

2.1 Image segmentation

In our system, the query image and all images in the database are first segmented into regions. The fuzzy features of color, texture, and shape are extracted to be the signature of each region in one image. The image segmentation is based on the color and spatial variation features using the k-means algorithm [18]. We use this algorithm to perform the image segmentation because it is unsupervised and efficient, which is crucial for segmenting general-purpose images such as the images on the World Wide Web. To segment an image, the system first partitions the image into blocks with 4×4 pixels to compromise between texture effectiveness and computation time, and then extracts a feature vector consisting of six features from each block. Three of them are average color components in a 4×4 pixel-size block. We use the *LAB* color space due to its desired property that the perceptual color difference is proportional to the numerical difference. These features are denoted as $\{C_1, C_2, C_3\}$. The other three features represent the energy in the high-frequency bands of the Haar wavelet transform [10], that is, the square root of the second-order moment of the wavelet coefficients in the high-frequency bands. To obtain these moments, a Haar wavelet transform is applied to the L component of each pixel. After a one-level wavelet transform, a 4×4 block is decomposed into four frequency bands; each band contains 2×2 coefficients. Without loss of generality, suppose that the coefficients in the *HL* band are $\{c_{k,l}, c_{k,l+1}, c_{k+1,l}, c_{k+1,l+1}\}$. Then we compute one feature of this block in the *HL* band as:

$$f = \sqrt{\frac{1}{4}\sum_{i=0}^{1}\sum_{j=0}^{1} c_{k+i,l+j}}\,. \tag{1}$$

The other two features are computed similarly from the *LH* and *HH* band. These three features of the block are denoted as $\{T_1, T_2, T_3\}$. They can be used to discern texture by showing L variations in different directions. Figure 2 illustrates the image partition and texture feature extraction procedure. After we have obtained feature vectors for all blocks, we perform normalization on both color and texture features to whiten them such that the effects of different

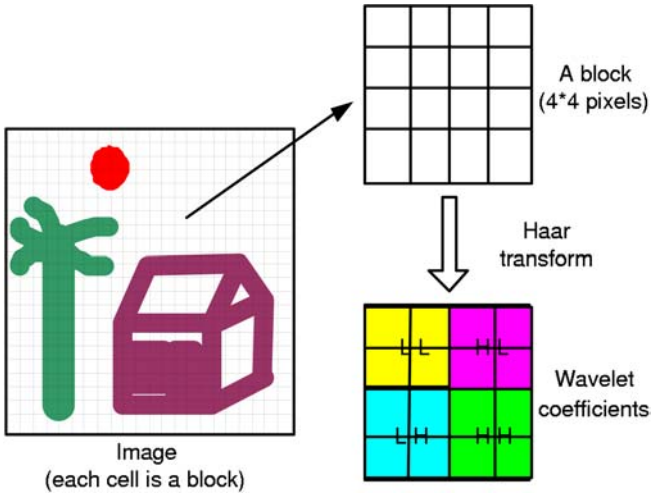

Fig. 2 Image partition and texture feature extraction

feature ranges are eliminated. Then the k-means algorithm [18] is used to cluster the feature vectors into several classes with each class corresponding to one region in the segmented image. Since the clustering is performed in the feature space, blocks in each cluster do not necessarily form a connected region in the image. Consequently, we preserve the natural clustering of objects in general-purpose images. The k-means algorithm does not specify how many clusters to choose. We adaptively select the number of clusters C by gradually increasing C until a stop criterion is met (see [46] for details of the stop criterion). The average number of clusters for all images in the database varies according to the different stop criteria. In the k-means algorithm we use, a color-texture weighted $L2$ distance metric

$$\sqrt{w_c \sum_{i=1}^{3} \left(C_i^{(1)} - C_i^{(2)}\right)^2 + w_t \sum_{i=1}^{3} \left(T_i^{(1)} - T_i^{(2)}\right)^2} \qquad (2)$$

is used to describe the distance between the features of blocks, where $C^{(1)}(C^{(2)})$ and $T^{(1)}(T^{(2)})$ are color features and texture features, respectively, of block 1(2). w_c and w_t are the weights specified in specific experiments. After the segmentation, three additional features are determined for each region to describe the shape property. They are normalized inertia [16] of order 1–3. For a region H in the 2D Euclidean integer space $\mathbb{Z}^2$ (an image), its normalized inertia of order p is

$$l(H, p) = \frac{\sum_{(x,y):(x,y)\in H}[(x - \hat{x})^2 + (y - \hat{y})^2]^{p/2}}{[V(H)]^{1+p/2}}, \qquad (3)$$

where $V(H)$ is the number of pixels in region H and $(\hat{x}, \hat{y})$ is the centroid of H. The minimum normalized inertia is achieved by spheres. Denoting the pth-order normalized inertia of spheres as L_p, we define the following three features to describe the shape of each region:

$$S_1 = l(H, 1)/L_1, S_2 = l(H, 2)/L_2, S_3 = l(H, 3)/L_3. \qquad (4)$$

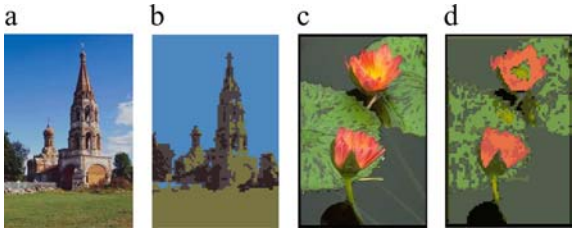

Fig. 3 Regions obtained for two exemplary images; each region is labeled with the average color of blocks in it. **a,c** Original images. **b,d** Segmented images (four regions in **b** and seven regions in **d**)

Figure 3 shows the segmentation results for two exemplary images. In this figure, panels a and c are two images in the database, and panels b and d are their region representations, respectively. Each region segmented is labeled by the average color of all the blocks associated with the region. As noted, four regions were obtained for image 3a and seven regions were obtained for image 3b.

2.2 Fuzzy color histogram for each region

The color representation would be coarse and imprecise if we simply extracted the block color feature to index each region as Wang et al. [44] proposed. Color is one of the most fundamental properties to discriminate images, so that we should take advantage of all available information in it. Considering the typical uncertainty stemming from color quantization and human perception, we develop a modified color histogram using the fuzzy technique [30, 42] to accommodate the uncertainty.

The fuzzy color histogram is defined as follows. We assume that each color is a fuzzy set while the correlation among colors is modeled as a membership function of different fuzzy sets. A fuzzy set F on the feature space R^n is defined as a mapping $\mu_F : R^n \to [0, 1]$ with μ_F as the membership function. For any feature vector $f \in R^n$, the value of $\mu_F(f)$ is called the degree of membership of f to the fuzzy set F (or, in short, the degree of membership to F). A value closer to 1 for $\mu_F(f)$ means the feature vector f is more representative of the fuzzy set F than a feature vector g whose $\mu_F(g)$ is closer to 0.

An ideal fuzzy color model should have the resemblance inversely proportional to the intercolor distance. Based on this requirement, the most commonly used prototype membership functions include conic, trapezoidal, B-splines, exponential, Cauchy, and paired sigmoid functions [19]. We have tested the conic, trapezoidal, exponential, and Cauchy functions in FAST. In general, the performances[1] of the exponential and the Cauchy functions are better than those of the conic and trapezoidal functions. Considering the computational complexity, we use the Cauchy function due to its computational simplicity. The Cauchy function, $C : R^n \to$

[1] The performance means the average image retrieval accuracy given other settings of the system the same.

[0, 1], is defined as

$$C(\vec{x}) = \frac{1}{1 + \left(\frac{\|\vec{x} - \vec{v}\|}{d}\right)^{\alpha}}, \tag{5}$$

where $\vec{v} \in R^n$, d and $\alpha \in R$, $d > 0$, $\alpha \geq 0$; $\vec{v}$ is the center location (point) of the fuzzy set; d represents the width of the function; and α determines the shape (or smoothness) of the function. Collectively, d and α describe the grade of fuzziness of the corresponding fuzzy feature.

Accordingly, the color resemblance in a region is defined as:

$$\mu_c(c') = \frac{1}{1 + \left(\frac{d(c,c')}{\sigma}\right)^{\alpha}}, \tag{6}$$

where d is the Euclidean distance between color c and c' in the *LAB* space and σ is the average distance between colors:

$$\sigma = \frac{2}{B(B-1)} \sum_{i=1}^{B-1} \sum_{k=i+1}^{B} d(c, c'), \tag{7}$$

where B is the number of bins in the color partition. The average distance between colors is used to approximate the appropriate width of the fuzzy membership function. The experiments show that, given the same other settings of the system, the average retrieval accuracy changes insignificantly when α is in the interval [0.7, 1.5] but degrades rapidly outside the interval. We set $\alpha = 1$ in Eq. 6 to simplify the computation.

This fuzzy color model enables us to enlarge the influence of a given color to its neighboring colors according to the uncertainty principle and the perceptual similarity. This means that each time a color c is found in an image, it influences all the quantized colors according to their resemblance to the color c. Numerically, this could be expressed as:

$$h_2(c) = \sum_{c' \in U^C} h_1(c') \mu_c(c'), \tag{8}$$

where U^C is the color universe in the image and $h_1(c')$ is the normalized conventional color histogram. Finally, the normalized fuzzy color histogram is computed as:

$$h(c) = \frac{h_2(c)}{\max_{c' \in U^C} h_2(c')}, \tag{9}$$

which maps to the range [0, 1].

Note that this fuzzy histogram operation is in fact a linear convolution between the conventional color histogram and the fuzzy color model. This convolution expresses the histogram smoothing, provided that the color model is indeed a smoothing, low-pass filtering kernel. The use of the Cauchy function as the color model produces the smoothed histogram, which is a means to reduce the quantization errors [21].

In the FAST prototype implementation, the *LAB* color space is quantized into 96 bins[2] by using uniform quantization (L by 6, A by 4, and B by 4). To reduce the online computation, for each bin $\mu_c(c')$ is precomputed and implemented as a lookup table.

2.3 Fuzzy representation of texture and shape for each region

To accommodate the imprecise image segmentation and uncertainty of human perception, we propose to fuzzify each region generated in image segmentation using a parameterized membership function. The parameter for the membership function is determined based on the clustering results of the blocks. Like the color features, the fuzzification of the texture and shape feature vectors again brings a crucial improvement to the region representation of an image: fuzzy features naturally characterize the gradual transition between regions within an image. In our proposed representation scheme, a fuzzy feature set assigns weights, called degrees of membership, to feature vectors of each block in the feature space. As a result, the feature vector of a block belongs to multiple regions with different degrees of membership as opposed to the classic region representation, in which a feature vector belongs to exactly one region. We first discuss the fuzzification of the texture features and then discuss that of the shape features.

We take each region as a fuzzy set of blocks. To propose a unified approach consistent with the fuzzy color histogram representation, we again use the Cauchy function to be the fuzzy membership function, i.e.,

$$\mu_i(f) = \frac{1}{1 + \left(\frac{d(f, \hat{f}_i)}{\sigma}\right)^{\alpha}}, \tag{10}$$

where $f \in R^k$ (here $k = 3$) is the texture feature vector of each block, $\hat{f}_i$ is the average texture feature vector of region i, d is the Euclidean distance between $\hat{f}_i$ and any feature f, and σ represents the average distance for texture features among the cluster centers obtained from the k-means algorithm. σ is defined as:

$$\sigma = \frac{2}{C(C-1)} \sum_{i=1}^{C-1} \sum_{k=i+1}^{C} \|\hat{f}_i - \hat{f}_k\|, \tag{11}$$

where C is the number of regions in a segmented image and $\hat{f}_i$ is the average texture feature vector of region i.

With this block membership function, the fuzzified texture property of region i is represented as:

$$\vec{f}_i^{\,T} = \sum_{f \in U^T} f \mu_i(f), \tag{12}$$

[2] Different numbers of bins are tested in FAST, namely, 96, 144, and 216 bins. The difference on the final image retrieval accuracy is negligible. For efficiency consideration, we use 96 bins.

where U^T is the feature space composed of texture features of all blocks.

Based on the fuzzy membership function $\mu_i(f)$ obtained in a similar fashion, we also fuzzify the shape property representation of region i by modifying Eq. 3 as:

$$l(i,p) = \frac{\sum_{f\in U^S}[(f_x-\hat{x})^2+(f_y-\hat{y})^2]^{p/2}\mu_i(f)}{[N]^{1+p/2}}, \tag{13}$$

where f_x and f_y are the x and y coordinates of the block with the shape feature f, respectively; $\hat{x}$ and $\hat{y}$ are the x and y central coordinates of region i, respectively; N is the number of blocks in an image; and U^S is the block shape feature space in an image. Based on Eqs. 4 and 13, we determine the fuzzified shape feature of each region, denoted as $x\ \vec{f}_i^S$.

2.4 Region matching and similarity determination

Once we have fuzzified representations for color, texture, and shape features, we apply the normalization on these features before they are written to the index files. For each region, we record the following information as its indexed data: (1) fuzzy color histogram $h(c)$, (2) fuzzy texture feature $\vec{f}^T$, (3) fuzzy shape feature $\vec{f}^S$, (4) relative size of region to whole image w, and (5) central coordinate of region area $(\hat{x},\hat{y})$. Such indexed data of all regions in an image are recorded as the signature of the image.

Based on the fuzzified features for regions in every image a fuzzy matching scheme is developed to determine the distance between any two regions p and q as well as the overall similarity between images. For fuzzy texture and shape features, we apply the $L2$ distance formula as:

$$d_T^{pq} = \|\vec{f}_p^T - \vec{f}_q^T\|$$

and

$$d_S^{pq} = \|\vec{f}_p^S - \vec{f}_q^S\|,$$

respectively.

For fuzzy histograms, we use the distance formula

$$d_C^{pq} = \sqrt{\frac{\sum_{i=1}^{B}[h_p(i)-h_q(i)]^2}{B}}, \tag{14}$$

where B is the number of bins and $h_p(i)$ and $h_q(i)$ are the fuzzy histograms of regions p and q, respectively.

The intercluster distance on color and texture between regions p and q is defined as:

$$d_{CT}^{pq} = \sqrt{d_C^{pq\,2} + d_T^{pq\,2}}. \tag{15}$$

The overall distance between two regions is defined as:

$$\mathrm{DIST}(p,q) = w d_{CT}^{pq} + (1-w) d_S^{pq}, \tag{16}$$

where w is a weight. In the current FAST prototype, we set $w = 0.7$ to purposely give color and texture more weight than shape, as we have found that shape features are more vulnerable to segmentation.

Since image segmentation is usually not perfect, a region in one image could correspond to several regions in another image. Consequently, we construct an image distance measure through the following steps. Suppose that we have M regions in image 1 and N regions in image 2.

Step 1: Determine the distance between one region in image 1 and all regions in image 2. For each region i in image 1, the distance between this region and image 2 is:

$$R_{i,\mathrm{Image2}} = \min_j \mathrm{DIST}(i,j), \tag{17}$$

where j is each region in image 2.

Step 2: Similarly, we determine the distance between a region j in image 2 and image 1:

$$R_{j,\mathrm{Image1}} = \min_i \mathrm{DIST}(j,i), \tag{18}$$

where i is each region in image 1.

Step 3: After obtaining the $M+N$ distances, we define the distance between the two images (1 and 2) as:

$$\mathrm{DistIge}(1,2) = \frac{\sum_{i=1}^{M} w_{1i} R_{i,\mathrm{Image2}} + \sum_{j=1}^{N} w_{2j} R_{j\mathrm{Image1}}}{2}, \tag{19}$$

where w_{1i} is the weight for each region in image 1 and w_{2j} is the weight for each region in image 2. We set $w_{1i} = \frac{N_{1i}}{N_1}$, where N_{1i} is the number of blocks in region i and N_1 is the total number of blocks in image 1. w_{2j} is defined similarly. In other words, larger regions are given more weight than smaller regions because we believe that larger regions are more semantically related to the content of an image. Clearly, $\mathrm{DistIge}(1,2) = 0$ if images 1 and 2 are identical, and $\mathrm{DistIge}(1,2)$ becomes larger when images 1 and 2 differ more substantially. Consequently, for each query q, $\mathrm{DistIge}(q,d)$ is determined for each image d in the database and relevant images are retrieved through sorting the similarities $\mathrm{DistIge}(q,d)$.

3 Hierarchical indexing structure and HEAR online search

To achieve fast retrieval, we have designed a hierarchical indexing structure in the database and a related online search algorithm to avoid a linear search. An optimal indexing structure is defined in the region space such that a query image only needs to be compared with those in the database that have at least one region that is most similar to a region in the query image given a specified similarity.

Let S denote the set of all the nodes in the indexing structure and X be the set of all the regions in the database. Each node $s \in S$ is a set of regions $X_s \subset X$ with a feature vector z_s, the centroid of the region feature set F_s in the node. The

children of a node $s \in S$ are denoted as $c(s) \subset S$. The child nodes partition the region space of the parent node such that

$$X_s = \bigcup_{r \in c(s)} X_r \,. \tag{20}$$

Now the question is how to construct such an optimal indexing structure. Recall that we have used a modified k-means algorithm in the image segmentation to form all the regions. After all the images in the database are indexed based on the indexing scheme in Sect. 2, we apply the same k-means algorithm again to all the feature vectors corresponding to all the regions of all the images recursively to form the hierarchy of the indexing structure. All the nodes represent centroid feature vectors of a corresponding set of regions except for the leaf nodes, which, in addition to a set of regions belonging to the corresponding cluster, also record IDs of images that share one region with the feature vectors of the regions in that set. The depth of the indexing structure is determined adaptively based on the size of the image database. The resulting indexing tree is called the *hierarchical* indexing structure.

Typical search algorithms would traverse the tree top-down, selecting the branch that minimizes the distance between a query q and a cluster centroid z_s . However, this search strategy is not optimal since it does not allow backtracking. To achieve an optimal search, we apply A^* search algorithm [36] by keeping track of all nodes that have been searched and always selecting the nodes with the minimum distance to the query region. The A* search is guaranteed to select the cluster whose centroid has the minimum distance in the set of visited nodes to the query region. Hence, it is optimal.

Note that in general the image set associated with a leaf node is *significantly* smaller than the original image set in the database. We show that this reduced image set may be further filtered in the online query search by exploiting the triangle inequality principle. Recall that the similarity function between two regions defined in Eq. 16 is metric, given two regions p and q associated with two images in the image set of a leaf node in the indexing tree. We have:

$$\mathrm{DIST}(p, q) \geq |\mathrm{DIST}(p, z) - \mathrm{DIST}(q, z)| \,, \tag{21}$$

where $\mathrm{DIST}(p, q)$ is the distance between regions p and q and z denotes a key region feature represented by the centroid of the corresponding leaf node (cluster). Consider a set of I regions $X_h = \{x_{h1}, x_{h2}, \ldots, x_{hI}\}$ at leaf node h and a key region feature z_h. Precalculating $\mathrm{DIST}(x_{hi}, z_h)$, for $i = 1$ to I, results in a linear table of I entries. In order to find those regions $x \in X_h$ at node h such that $\mathrm{DIST}(r, x) \leq t$ for a query region r and the predefined threshold t, we note that the lower bounds on $\mathrm{DIST}(r, x)$ exist by determining $\mathrm{DIST}(r, z_h)$, $\mathrm{DIST}(x_{hi}, z_h)$ and repeatedly applying Eq. 21. If $|\mathrm{DIST}(r, z_h) - \mathrm{DIST}(x_{hi}, z_h)| > t$, then x can be safely eliminated from the linear table of I entries, obviating the need to search for all entries in the table. Thus, given a query, we have the following retrieval algorithm called the hierarchical, elimination-based A* retrieval (HEAR) algorithm and a theorem guaranteeing the logarithm complexity in average case performance for HEAR.

The symbols used in the HEAR algorithm are introduced as follows. m is the number of regions in the query image, s^* is the cluster whose centroid has the minimum distance to the query region r, Ω is the cluster set we have searched, $|c(s^*)|$ is the size of the child set of s^*, z_s is the cluster centroid, *NodesSearched* records the number of nodes searched so far, and t is the predefined threshold of the distance between a region and a query region. The resulting Ψ is the final image set to be compared with the query image.

```
input  : q, the query image
output : Ψ, Images retrieved for the query image q
begin
    for For each region r in the query image q do
        s* = root;
        Ω = {s*};
        NodesSearched = 0;
        while s* is not a node of desired tree depth do
            Ω ← (Ω − {s*}) ∪ c(s*);
            NodesSearched = NodesSearched + |c(s*)|;
            s* ← arg min_{s∈Ω}(DIST(r, z_s));
        end
        Φ = {};
        for each region p in the node s* do
            if |DIST(p, z_s) − DIST(r, z_s)| ≤ t then
                Φ ← Φ ∪ {p};
            end
        end
        Ψ_r ={Images having regions in set Φ};
    end
    Ψ = ⋃_{r=1}^{m} Ψ_r;
end
```

Algorithm 1: HEAR Algorithm

Theorem 1 *In the average case, the HEAR algorithm achieves logarithm retrieval efficiency.*

Proof Suppose that m is the average branching factor of the hierarchical indexing structure, n is the number of images in the database, l is the average number of regions of an image, and k is the height of the indexing tree. Then nl is the total number of regions. In the average case, it is clear that when $k \geq \log_m nl - \log_m(\log_m nl)$, the number of regions in a leaf node $w \leq \log_m nl$. In the selected leaf node s^*, the triangle inequality principle is applied. Without loss of generality, suppose that the distance threshold ratio between a region and the centroid for the region to be selected as a candidate region is $1/\tau$. Consequently, the average number of regions selected to compare with the query region is $q \propto w/\tau^2 = \frac{\log_m nl}{\tau^2}$. We call these regions candidate regions. Each candidate region corresponds to one image in the database. Thus, the total number of images in the database to be compared with the query image is $\lambda lq = \frac{\lambda l \log_m nl}{\tau^2}$, where λ is the ratio that describes the region-to-image correspondence

relationship, $\lambda \in [1/l, 1]$. Thus we observe that the average number of images to be compared is bounded in $[\frac{\log_m nl}{\tau^2}, \frac{l \log_m nl}{\tau^2}]$. l is determined by the resolution of the image segmentation and is typically small (4 in the FAST prototype). τ is a constant. Hence the complexity of the online search algorithm HEAR is $O(log_m n)$ for a database of n images. □

While any feature-based CBIR method could apply a clustering algorithm recursively to generate a hierarchy in the (typically high-dimensional) feature space, we argue that this does not work in general and thus show that the contributions reflected in Theorem 1 are unique and significant.

We define a classification-based clustering in a feature space as being *spherically separable* [41] if for any cluster there is always a specific radius R for this cluster such that for any feature vector v, v is in the cluster if and only if $D(v, c) < R$, where c is the centroid vector of the cluster and D is a metric distance measure. Given a CBIR method, if the similarity measure is metric, and if the images are indexed in global features (i.e., each image is indexed by a single feature vector in the image feature space), in order to generate a hierarchy in the feature space by recursively clustering the features of the whole image database, it would require that all the clusters be spherically separable. Clearly this is not true, as in a typical image feature space the distribution of the semantic classes in the feature space is rather complicated (e.g., it is typical that one semantic class is contained by another, or two semantic classes are completely "mixed" together), and thus the spherically separable property is by no means satisfied. This is shown to be a well-known fact even in many special domain image classification or clustering problems such as in face image classification [4], not to mention in general domain image retrieval. On the other hand, if the similarity measure is not metric, it would not be possible to generate a hierarchy in a feature space based on the recursive applications of a clustering algorithm as the clustering presumes a metric distance measure. Consequently, the only possibility to generate such an indexing hierarchy is to use a nonglobal feature, i.e., to build up this hierarchy in a feature space other than the image feature space. The significance of the development of the hierarchical indexing structure as well as the related HEAR online search algorithm reflected through Theorem 1 is that we have explicitly developed an indexing scheme in the *regional feature space* and have shown that even with this "detour" through the regional feature space we can still promise a logarithm search complexity in the average case in the image feature space.

We develop the hierarchical indexing structure in the regional feature space based on the following three reasons. First, since the features we have developed to index images are based on the regional feature space, it is natural to build up an indexing hierarchy in the regional feature space. Second, the similarity measure defined in FAST is not metric, and thus it is not possible to directly build up an indexing hierarchy in the image feature space. Third, after segmentation of an image into regions, the features in the regional feature space are essentially "uniform," and consequently they are able to satisfy the spherically separable property in the regional feature space, which is required for the construction of a hierarchy using clustering.

Apart from the above discussions of the hierarchical indexing structure, it is also interesting to compare it with the existing high-dimensional indexing structures, e.g., R-tree and its derivative indexing trees. It has been demonstrated that the search efficiency of an R-tree is largely determined by the coverage and overlap [24]. The coverage of a level of an R-tree is the total area of all the rectangles associated with the nodes of that level. The overlap of a level of an R-tree is the total area contained within two or more nodes. Efficient R-tree search demands that both the coverage and overlap be minimized. From the point of view of R-tree-based multidimensional access structures, the proposed hierarchical indexing structure has two advantages. First, the nodes in each level have no overlap since each region feature belongs to only one cluster (node). With this property, multiple-path traversal is avoided, which improves the search efficiency significantly. Second, the search on the hierarchical indexing structure does not depend on the node coverage because no dimension comparisons are required to decide the branch in HEAR. In other words, the minimum bounding region (MBR) of each internal node in the hierarchical indexing structure is determined by the region features per se and can be any possible shape. With the nonoverlapping property between internal nodes of each level and MBR-coverage-independent search in HEAR, the efficiency of the hierarchical indexing structure is enhanced substantially as compared with the discussed R-tree as well as its derivative data structures for region-based CBIR.

4 Addressing user subjectivity using ITP and ARWU

To achieve semantics-tailored retrieval, we must address the *human perception subjectivity* [35] issue in CBIR. This is resolved through the ITP and ARWU algorithms we have developed with user query preference inferred to be able to deliver individualized retrieval.

Since the relevance subjectivity in FAST resides at the region level as opposed to at the image level, ideally we would like to ask users to indicate the relevant regions in each retrieval, which would add complexity to the user interface and user interaction. As a compromise, FAST assumes that users only cast yes (+) or no (−) votes to each retrieved image as the data collected in the user relevance feedback. Since they are based on this very "qualitative" user relevance feedback, the feedback data are typically sparse and limited. We have developed an algorithm to infer user preference in order to tailor to the intended retrieval from a sparse distribution of these "qualitative" data. Moreover, we take advantage of this user relevance feedback information to further expedite the subsequent query search, resulting in achieving the two goals of fast and semantics-tailored retrieval simultaneously.

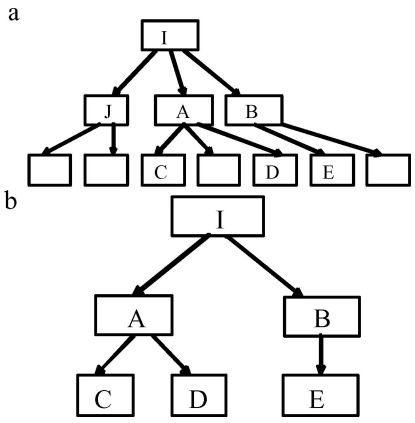

Fig. 4 Example of an original index tree and one session tree after pruning using ITP. **a** Original Indexing Tree (lettered leaf nodes are relevant and blank leaf nodes are irrelevant, which are derived from the hyperplane function H). **b** The session tree after pruning

We note the fact that the similar (common) regions among relevant images are important to characterize relevant images, whereas the similar (common) regions among irrelevant images are important to distinguish the retrieved irrelevant images from the relevant images [35]. Assuming that a user's personal preference for the intended retrieval is consistent over the whole session of the retrieval, we develop a user relevance feedback algorithm called indexing tree pruning (ITP). The idea of ITP is that we use the k-means algorithm to infer the "typical" regions from the images voted as relevant, and the "typical" regions from the images voted as irrelevant, based on which a standard two-class support vector machine (SVM) [41] is used to generate a separation hyperplane in the region feature space, which in turn "cuts" the space into two halves; the subsequent search may be further constrained to focus on the relevant side of the hierarchical indexing structure using HEAR. Specifically, ITP is listed in Algorithm 2.

Figure 4 illustrates an example of the original indexing tree and one session tree after pruning using ITP. The actual pruning is done through applying the DBT algorithm [22]. Note that ITP differs from the existing SVM-based user relevance feedback algorithms, such as [6], which typically require a large number of voted samples to obtain a classifier in the feature space. In ITP, the SVM is not used directly to perform image retrieval; instead, it is used to guide a coarse filtering (pruning) such that the hierarchical indexing structure in the database can be tuned in favor of the user's relevancy intention. In Sect. 5, we shall see that with a relatively small number of leaf nodes ($<$1000) and reasonable dimensionality of a feature space (9 in the indexing scheme), a relatively small number (15–30) of voted images can boost the performance well and further expedite the subsequent retrieval at the same time.

While ITP is able to infer relevancy and irrelevancy from the voted images, we make a further effort in attempting to infer the degree of relevancy or irrelevancy. Following the "most similar, highest priority (MSHP)" principle [44], we can adaptively update the region weights in Eq. 19 based on the feedback data. We implement this idea and call it the adaptive region weight updating (ARWU) algorithm.

input : Images users labeled
output : Ψ, Images retrieved after learning
begin
  Initialization, set $RegR = \{\}$; $RegI = \{\}$;
  Applying the modified k-means clustering algorithm on the region subspace consisting of relevant images, m clusters are obtained. They are sorted in terms of their number of regions, denoted as $SR = \{R_1, R_2, \ldots, R_m\}$, where $\|R_i\| \geq \|R_j\|$ for $i < j$;
  Applying the modified k-means clustering algorithm on the region subspace consisting of irrelevant images, n clusters are obtained. They are sorted in terms of their number of regions, denoted as $SI = \{I_1, I_2, \ldots, I_n\}$, where $\|I_i\| \geq \|I_j\|$ for $i < j$;
  $RegR \leftarrow RegR \cup R_k, k = 1, \ldots, m$;
  $RegI \leftarrow RegI \cup I_k, k = 1, \ldots, n$;
  A Gaussian RBF-based two-class SVM is applied on the relevant and irrelevant region set $RegR$ and $RegI$ to learn the hyperplane H, which separates the region space into relevancy and irrelevancy parts;
  $Y = \{\}$; $X = \{\}$;
  for *the centroid of each leaf node in the indexing tree,* t_i **do**
    Determine $H(t_i)$;
    if $H(t_i) \geq 0$ **then**
      $Y \leftarrow Y \cup \{t_i\}$;
    else
      $X \leftarrow X \cup \{t_i\}$;
    end
  end
  Pruning the indexing tree, reserving only ancestor nodes of Y to generate a session tree ST;
  Performing online search algorithm HEAR on the session tree ST, return result Ψ;
end

Algorithm 2: ITP Algorithm

The idea of ARWU is as follows. The cardinality of a cluster to which a query region belongs in the relevant region space is an indicator of the commonality of this region to the relevant image set. With the voted relevant and irrelevant images, for every region in the query image, the weights of the regions that are similar to the regions in the relevant images but dissimilar to the regions in the irrelevant images should increase, and otherwise the weights should decrease. For regions in each target image (i.e., image to be compared with the query image), a high weight is given to regions with a smaller distance to the query image, and in the meantime the weight is adjusted to a higher value if it is the most similar to a query region with a high weight already. Otherwise, the weight of the region is lowered accordingly. Consequently, more weight is given to regions on which the user's query intention, learned from the feedback examples by ARWU, focuses. The nature of this weight adjustment algorithm is a discriminant whitening transform learned from both inferred relevant and irrelevant regions. In addition, the weights adjusted still preserve a desired characteristic for the distance metric, i.e., *the distance between two identical images equals 0*. ARWU is used in conjunction with ITP for further improving semantics-tailored retrieval. ARWU is

listed in Algorithm 3. In the algorithm, η_i acts as a penalty, reflecting the effect of negative examples to each region in the query image.

input : q, the query image
output : Updated region weights
begin
  for *each region in the query image,* Q_i, i=1,...,M **do**
    the cluster containing it in the relevant image set is obtained as C_i, $C_i \in SR$, the cluster containing it in the irrelevant image set is obtained as D_i, $D_i \in SI$;
    $WR_i = \frac{\|C_i\|}{\sum_k \|R_k\|}$;
    $WI_i = \frac{\|D_i\|}{\sum_k \|I_k\|}$;
    $W_{1i} = \frac{\eta_i * WR_i}{\sum_{k=1}^{M} (\eta_k * WR_k)}$, where $\eta_i = 1 - WI_i$;
  end
  for *each region in a target image,* T_j, j=1,...,N, *its most similar region in the query image is denoted as* S_j **do**
    $U_j = \frac{W_{1S_j}}{R_{j,Image1}}$, where $R_{j,Image1}$ is the distance between this region, T_j, and the query image, which is defined in Eq. 18;
    The weight for each region is normalized as $W_{2j} = \frac{U_j}{\sum_{k=1}^{N} U_k}$;
  end
  Applying W_{1i} and W_{2j} to Eq. 19 to determine the distance between the query image and each target image;
end

Algorithm 3: ARWU Algorithm

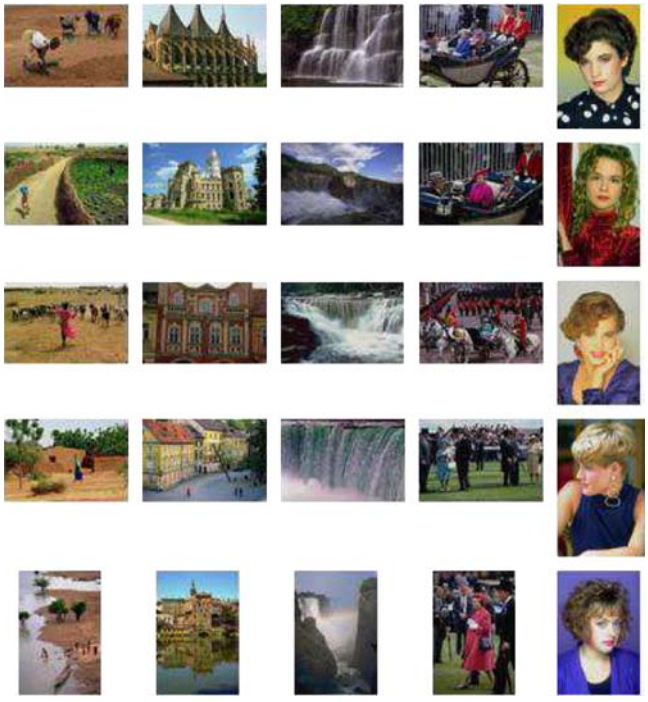

Fig. 5 Sample images in the testing database. The images in each column are assigned to one category. From *left* to *right*, the categories are "African rural area," "historical building," "waterfalls," "British royal event," and "model portrait"

5 Experimental evaluations

We have implemented the FAST methodology in a prototype system. For discussion and reference purposes, we also call the prototype FAST. The following reported evaluations are performed in a general-purpose color image database containing 10,000 images from the COREL collection of 96 semantic categories, including people, nature scene, building, and vehicles. No prerestrictions on camera models, lighting conditions, etc. are specified in the image database for the testing. These images are all in JPEG format. We choose this database to test the FAST methodology because it is available to the public and is used in the evaluations of several state-of-the-art CBIR systems, e.g., IRM [43] and UFM [7]. The database is accessible at http://www.fortune.binghamton.edu/download.html. Figure 5 shows several samples of the images belonging to a few semantic categories in the database. Each semantic category in this image database has 85 to 120 associated images. From this database 1,500 images are randomly selected from all the categories as the query set. A retrieved image is considered a match if it belongs to the same category of the query image. We note that the category information in the COREL collection is only used to truth the evaluation; we do not make use of this information in the indexing and retrieval.

5.1 System implementation

The FAST prototype is implemented on a Pentium III 800-MHz computer with 256 MB memory. After performing the image segmentation described in Sect. 2.1, the homogenous regions of each image are obtained. The segmentation results indicate that the regions extracted are related to the objects embodying image content. In the evaluation, a total of 56,722 regions are extracted for all 10,000 images in the database, which means on average 5.68 regions are extracted in each image. Image segmentation for the testing database takes 5.5 h to complete, corresponding to about 1.9 s for each image.

For each image, the fuzzy color, texture, and shape features are determined for each region. Based on these features of all regions extracted for the database, the hierarchical indexing structure of four levels is constructed offline. All regions are partitioned into several classes using the modified k-means algorithm. In this evaluation, the total number of classes is determined to be 677, with the maximum number of regions in one class being 194 and the minimum number of regions in one class being 31. For each class, a hash table mapping between the associated regions and the corresponding image names in the database is maintained. The generation of the four-level hierarchical indexing structure takes 70 min. While the entire indexing process is offline, the online query processing is fast. On average, the query time for returning the top 30 images is less than 1 s.

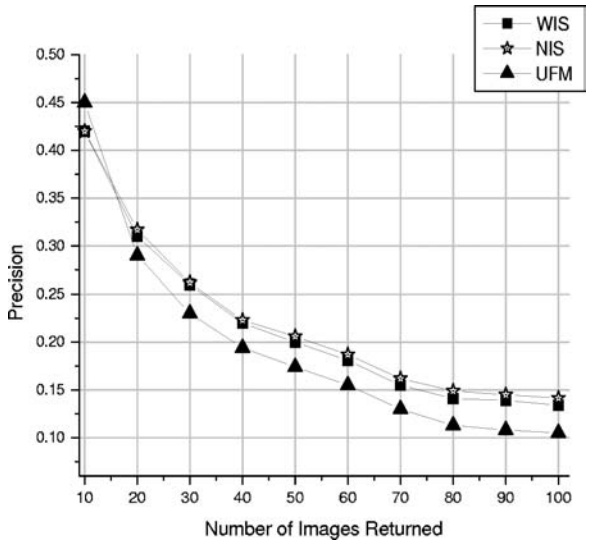

Fig. 6 Average precision/scope comparisons between two versions of FAST (WIS and NIS) and UFM

5.2 Evaluation of retrieval accuracy and scalability

FAST is evaluated against one of the state-of-the-art CBIR systems, UFM [7], with respect to effectiveness. Retrieval effectiveness is measured by recall and precision metrics [1]. For a given query and a given number of images retrieved, precision gives the ratio between the number of relevant images retrieved and the number of retrieved images in total. Recall gives the ratio between the number of relevant images retrieved and the total number of relevant images in the collection.

To test the effects of the hierarchical indexing structure and the HEAR algorithm, we have ported FAST into two separate versions: one with the original FAST methodology (which uses the hierarchical indexing structure in the region feature space and the HEAR online search), called WIS, and the other with the hierarchical indexing structure and the HEAR disabled, i.e., using the FAST indexing scheme in a linear search, called NIS. Both versions of FAST are compared with UFM in the 10,000-image database. The precision-scope data are recorded in Fig. 6, which demonstrates that the retrieval effectiveness of FAST is in general superior to that of UFM and the application of the hierarchical index structure and the HEAR algorithm does not degrade the performance perceivably; in fact, the performance of NIS is always about the same as that of WIS.

The hierarchical indexing structure of FAST is generated as a tree with a depth of 4 and average branching factor of 5. The tree is almost balanced, which confirms that the clustering in the region feature space based on FAST methodology is almost spherically separable [41]. This configuration is a tradeoff between the recall and precision. In FAST, each node of the tree is implemented as an object serialized to a physical file on the disk. Thus parallel searching on the indexing tree is made possible (which is an expected improvement for the future further development of FAST). We set the threshold for the triangle-inequality comparison of the region distance in all the leaf nodes as an adjustable parameter, which can be set by the users. Given the normalized region feature distance in [0, 1], the current value of the threshold in FAST is 0.3.

To study the scalability of FAST, we incrementally sample the original 10,000-image database to generate two smaller databases, one with 3,000 images and the other with 6,000 images. These two databases contain sampled images from all 96 categories. Consequently, the depths of the hierarchical indexing structures for the two databases are set to be $\lfloor \log_5 nl - 3 \rfloor$ accordingly based on the same principle used for the original image database, resulting in 2 and 3, respectively. We randomly sample 100 images to form a query set from each of the three databases. For the query images the average number of images compared in each of the three databases, the average indexing structure search overhead (the processing time to traverse the hierarchical indexing structure for searching the pursued leaf node), the average query processing time in FAST, and the average query processing time in linear search are documented in Table 1. The table shows that the average number of compared images is significantly reduced in comparison with the database size. In addition, as indicated in Table 1, although the hierarchical indexing structure search introduces the computation overhead, the average total query processing time is still much less than the average query processing time in the linear search due to the reduced number of images to be compared with. The computation overhead for the hierarchical indexing structure search is small because only a few additional distance computations are performed and highly efficient hash table searches are applied. With the increase of the database size, the percentage of the images examined and the average computation overhead remain relatively stable. The average query processing time is much less than that in the linear search in all three testing databases. The average efficiency improvement on the query processing time to the linear search is 72.7%. This result, combined with the results observed in Fig. 6,

Table 1 Retrieval efficiency results; percentage of images compared in the three databases

Database size	Average # of compared images	Average percentage of images examined (%)	Average search overhead in FAST (s)	Average query processing time in FAST (s)	Average query processing time in linear search (s)
3,000	795	26.5	0.08	0.55	1.78
6,000	1032	17.2	0.12	0.79	3.04
10,000	1140	11.4	0.15	0.98	3.96

indicates the promise of FAST for efficiently handling large image databases without sacrificing retrieval effectiveness.

Since in FAST the size of the class level (clusters in the region feature space) information is much smaller than that of the index files for images in the database (in our experiments, the size ratio is 1/95–1/120), it is practical and desirable to deposit the class level information in the main memory. With such a design the I/O costs for each query are only proportional to the number of images compared. The reduced I/O costs in the FAST query processing are observed as shown in Table 1 as well.

5.3 Evaluation of the retrieval robustness

In the indexing scheme we use the Cauchy function to correlate color descriptions and to smooth the regions (equivalent to a convolution) so that the color alteration and segmentation inaccuracy issues are addressed explicitly. On the other hand, since image segmentation is always subject to error, being robust to segmentation uncertainties becomes a critical performance index for a region-based image retrieval system. To evaluate the effectiveness of the fuzzified features for improving the robustness to color variations and segmentation uncertainties, we compare the performances of FAST and UFM for color variations and image segmentation uncertainties. Color variations are simulated by changing colors to their adjacent values for each image, while the segmentation uncertainties in an image can be characterized by the entropy. For image i with C segmented regions, its entropy, $E(i)$, is defined as:

$$E(i) = -\sum_{j=1}^{C} P\left(R_j^i\right) \log\left[P\left(R_j^i\right)\right], \tag{22}$$

where $P(R_j^i)$ is the percentage of image i covered by region R_j^i. The larger the value of the entropy, the higher the uncertainty level. As we can see, the entropy $E(i)$ increases with the increase of the number of regions C. Thus we can adjust the uncertainty level by changing the value of C. C is controlled by modifying the stop criterion of the modified k-means algorithm described in Sect. 2.1.

For a fair comparison between FAST and UFM at different color variation and segmentation uncertainty levels, we perform the same evaluations for different degrees of color variation and average values of C (4.31, 6.32, 8.64, 11.62, and 12.25) on the 3,000-image database introduced above. To evaluate the robustness in the color variations we apply color changes to an image (target image) in the database. The modified image is then used as the query image, and the rank of the retrieved target image is recorded. Repeating the process for all images in the testing database, the average rank for target images are computed for FAST and UFM. The result is shown in Fig. 7. The average rank of the target image of FAST is lower than that of UFM for each level of color variation (in an acceptable range of color variation that does not affect semantic perception).

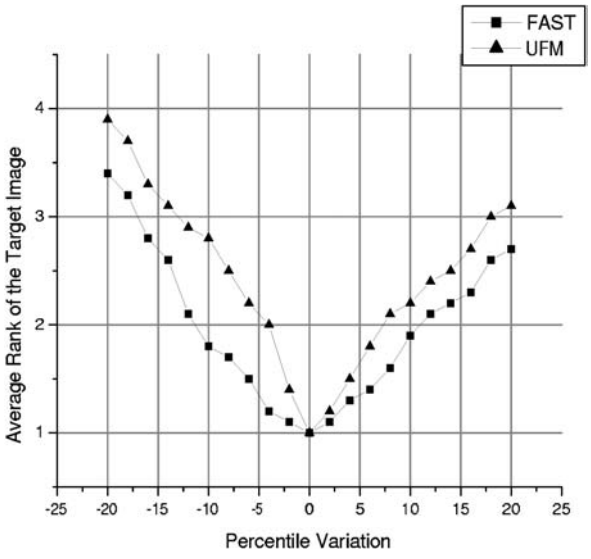

Fig. 7 Comparison of FAST indexing scheme and UFM method on robustness in color variation. Every image in the 3,000-image database is altered and used as a query image

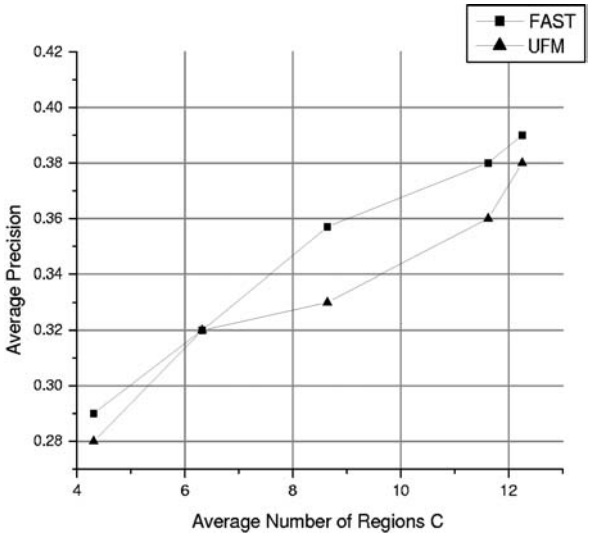

Fig. 8 Comparison of FAST indexing scheme and UFM method on robustness in image segmentation uncertainty

To evaluate the robustness in segmentation uncertainty, the performance in terms of overall average precision in the top 30 returned images is evaluated for both approaches. The result is given in Fig. 8. As we have shown, the entropy $E(i)$ (segmentation uncertainty) level increases when the image is segmented into more regions. At all segmentation uncertainty levels, FAST performs better than or camparable to UFM.

Combining these two robustness experiments, we have observed that the FAST indexing scheme is more robust than UFM with respect to color variation and segmentation uncertainty. The performance differences between FAST and UFM can be explained as follows. The UFM method uses

the representative feature (one feature vector) of each region to model the segmentation uncertainty, which is coarse and artificial. The model generated is not accurate enough to fit the segmented images well. On the other hand, the FAST indexing scheme leverages all block features in every region to generate fuzzy models for each feature component, and thus describes the segmentation uncertainty more precisely and effectively.

5.4 Evaluation of the effectiveness of ITP and ARWU

To evaluate ITP- and ARWU-based user relevance feedback, we invited a group of five users to participate in the evaluations. The participants consist of CS graduate students as well as laypeople outside the CS department. We asked different users to run FAST initially without the relevance feedback interaction, and then to place their relevance feedback after the initial retrievals. In each round of interaction, a user first checks relevant images as well as irrelevant images from the retrieval result and then invokes the next round of retrieval.

In FAST, the Gaussian kernel function used in SVM is $K(x, y) = e^{-\|x-y\|^2/2\sigma^2}$, with $\sigma = \sqrt{2}/2$. To evaluate the capabilities of algorithm ITP and ARWU, FAST is run on the 10,000-image database with the user relevance feedback mode for the 1,500-query-image set using a varied number of truncated top retrieved images. We plot the retrieval curves of the precision vs. the number of the top retrieved images in Fig. 9, which shows the average precision on three and five rounds of feedback. The semantics-tailored capability empowered by ITP and ARWU enhances the retrieval effectiveness in all the scenarios.

Another experiment is performed to verify the promise of FAST ITP itself on the basis of 100 query images. We record the average number of images compared in the first retrieval and after the third and fifth retrieval iterations in the three databases. The results are shown in Fig. 10.

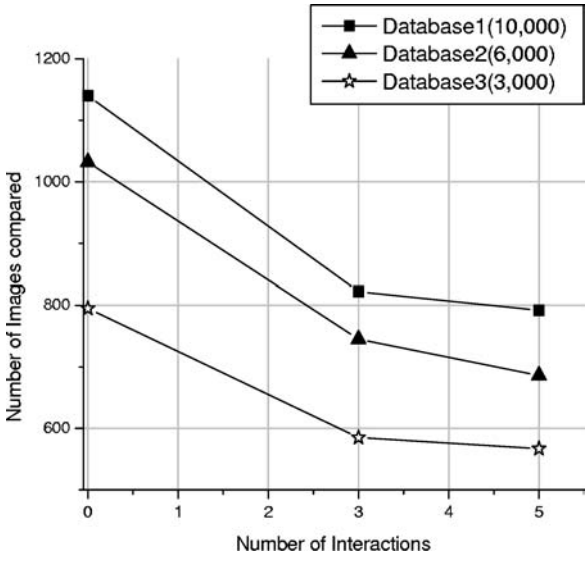

Fig. 10 Effect of ITP in iterations for the three databases

The reduction of the number of images compared in each iteration due to the tree pruning in ITP is observed. The efficiency boost between the third iteration and the initial retrieval is significant due to the relatively large portion of the leaf nodes pruned in the first three iterations. As the iterations progress, the ratio of nodes classified on the irrelevancy side of the SVM decreases quickly such that the efficiency boost decays accordingly.

To evaluate the effects of the ARWU algorithm for further enhancing semantics-tailored retrieval, we compare the average precision for the same 1,500-query-image set on the 10,000-image database, in the scenarios with and without running ARWU. The experiment is based on the average of the precisions for the top 30 returned images for each query. The results are shown in Fig. 11. As shown, the semantics-tailored retrieval effectiveness is improved substantially with ARWU.

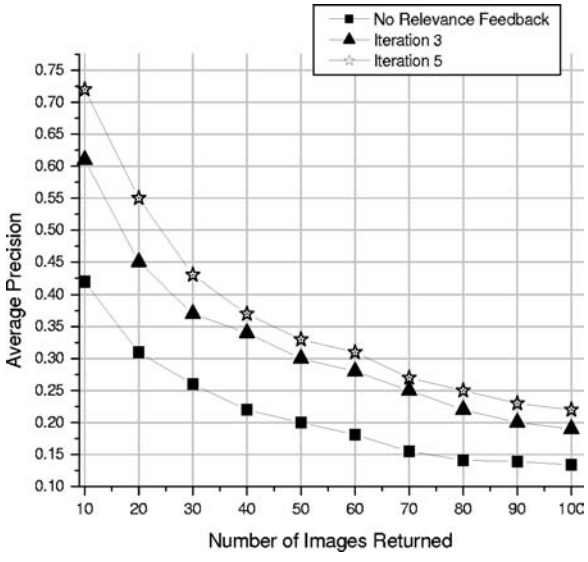

Fig. 9 Average precision vs. number of returned images with three and five rounds of user relevance feedback

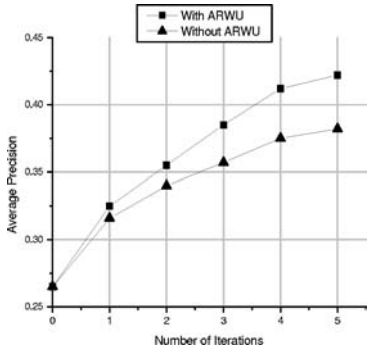

Fig. 11 Accuracy comparison when FAST ITP is run with and without ARWU algorithm

6 Conclusions

We have developed a new CBIR methodology based on the goal of delivering fast and semantics-tailored retrieval, and thus we call it FAST. FAST incorporates a new indexing scheme, a hierarchical indexing structure with a hierarchical, elimination-based A* online search algorithm called HEAR. We have shown that HEAR promises a logarithm search instead of linear search in an average case. FAST also not only offers user relevance feedback capability to address individualized retrieval based on user intention, but also takes advantage of the hierarchical indexing structure and the HEAR algorithm to achieve more effective and efficient semantics-tailored retrieval through applying the ITP and ARWU algorithms. The promise FAST demonstrates and the benefits FAST offers are sufficiently supported by the extensive experimental evaluations.

References

1. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, Reading, MA (1999)
2. Bentley, L.: Multi-dimensional binary search trees in database applications. IEEE Trans. Softw. Eng. **4**, 333–340 (1979)
3. Berman, A., Shapiro, L.G.: Efficient image retrieval with multiple distance measure. Proc. SPIE **3022**, 12–21 (1997)
4. Brunelli, R., Poggio, T.: Face recognition: features versus templates. IEEE Trans. Pattern Anal. Mach. Intell. **15**, 1042–1053 (1993)
5. Carson, C., Thomas, M., Belongie, e.a.S.: Blobworld: a system for region-based image indexing and retrieval. In: Proceedings of the 3rd International Conference on Visual Information Systems, pp. 509–516, Amsterdam (1999)
6. Chapelle, O., Haffner, P., Vapnik, V.N.: Support vector machines for histogram-based image classification. IEEE Trans. Neural Netw. **10**(5), 1055–1064 (1999)
7. Chen, Y., Wang, J.Z.: A region-based fuzzy feature matching approach to content-based image retrieval. IEEE Trans. Pattern Anal. Mach. Intell. **24**(9), 1252–1267 (2002)
8. Cox, I.J., Miller, M.L., Minka, T.P., Papathomas, T.V., Yianilos, P.N.: The bayesian image retrieval system, pichunter: theory, implementation and psychophysical experiments. IEEE Trans. Image Process. **9**(1), 20–37 (2000)
9. Dao, S., Yang, Q., Vellaikal, A.: $mb^+ - tree$: an index structure for content-based retrieval. In: Nwosu, K.C., Thuraisingham, B., Berra, P.B. (eds.) Multimedia Database Systems: Design and Implementation Strategies, Chap. 11. Kluwer, Norwell, MA (1996)
10. Daubechies, I.: Ten Lectures on Wavelets. Capital City Press, Montpelier, VT (1992)
11. Elmasri, R., Navathe, A.B.: Fundamentals of Database Systems, 2nd edn. Benjamin Cummings, Redwood City, CA (1994)
12. FJ et al.: An efficient region-based image retrieval framework. In: Proceedings of ACM Multimedia, Juan-les-Pins, France (2000)
13. MF et al.: Query by image and video content: the qbic system. IEEE Comput. **28**(9), 23–32 (1995)
14. NB et al.: The r^*-tree: an efficient and robust access mehtod for points and rectangles. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 322–331, Atlantic City, NJ (1990)
15. YK et al.: Knowing a tree from the forest: art image retrieval using a society of profiles. In: Proceedings of ACM Multimedia, Berkeley, CA (2003)
16. Gersho, A.: Asymptotically optimum block quantization. IEEE Trans. Inf. Theory **25**(4), 373–380 (1979)
17. Guttman, O.: R-tree: a dynamic index structure for spatial searching. In: Proceedings of ACM SIGMOD, pp. 47–57, Boston (1984)
18. Hartigan, J.A., Wong, M.A.: Algorithm as 136: a k-means clustering algorithm. Appl. Stat. **28**, 100–108 (1979)
19. Hoppner, F., Klawonn, F., Kruse, R., Runkler, T.: Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition. Wiley, New York (1999)
20. Huang, J., Kumar, S.R., Mitra M., Zhu W., Zabih R.: Image indexing using color correlograms. In: IEEE International Conference on Computer Vision and Pattern Recognition Proceedings, Puerto Rico (1997)
21. Kautsky, J., Nichols, N.K., Jupp, D.L.B.: Smoothed histogram modification for image processing. CVGIP: Image Understand. **26**(3), 271–291 (1984)
22. Kearns, Y., Mansour, M.J.: A fast, bottom-up decision tree pruning algorithm with near-optimal generalization. In: Proceedings of the 15th International Conference on Machine Learning, pp. 269–277, Madison, WI (1998)
23. Lin, K.I., Jagadish, H.V., Faloutsos, C.: The tv-tree: an index structure for high-dimensional data. VLDB J. **3**, 517–546 (1994)
24. Lu, G.: Techniques and data structures for efficient multimedia retrieval based on similarity. IEEE Trans. Multimedia **4**(3), 372–384 (2002)
25. Ma, W.Y., Manjunath, B.: Netra: a toolbox for navigating large image databases. In: Proceedings of the IEEE International Conference on Image Processing, pp. 568–571, Santa Barbara, CA (1997)
26. Marsicoi, M.D., Cinque, L., Levialdi, S.: Indexing pictorial documents by their content: a survey of current techniques. Image Vis. Comput. **15**, 119–141 (1997)
27. Minka, T.P., Picard, R.W.: Interactive learning using a society of models. Special Issue of Pattern Recognition on Image Database: Classification Retrieval **30**(4) (1997)
28. Nievergelt, J., Hinterbergre, H., Sevcik, K.C.: The grid file: an adaptive, symmetric multikey file structure. ACM Trans. Database Syst. **9**, 38–71 (1984)
29. Ogle, V., Stonebraker, M.: Chabot: retrieval from a relational databases of images. IEEE Comput. **28**(9), 40–48 (1995)
30. Pal, S.K., Ghosh, A., Kundu, M.K.: Soft Computing for Image Processing. Physica, Mostbach (2000)
31. Pass, G., Zabih, R.: Histogram refinement for content-based image retrieval. In: IEEE Workshop on Applications of Computer Vision. Sarasota, FL (1996)
32. Porkaew, K., Chakrabarti, K., Mehrotra, S.: Query refinement for multimedia similarity retrieval in mars. In: Proceedings of ACM Multimedia (1999)
33. Prabhakar, S., Agrawal, D., Abbadi, A.E.: Data clustering for efficient range and similarity searching. In: Proceedings of the SPIE Conference on Multimedia Storage and Archiving System III, vol. 3527, pp. 419–430, Boston (1998)
34. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest neighbor queries. In: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, pp. 71–79. San Jose (1995)
35. Rui, Y., Huang, T.S., Ortega, M., Mehrotra, S.: Relevance feedback: a power tool in interactive content-based image retrieval. IEEE Trans. Circuits Syst. Video Technol. **8**(5), 644–655 (1998)
36. Russell, S., Norvig, P.: Artificial Intelligence – A Modern Approach. Prentice Hall, Englewood Cliffs, NJ (1995)
37. Samet, H.: Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS. Addison-Wesley, Reading, MA (1989)
38. Sellis, T., Roussopoulos, N., Faloutsos, C.: The r^+-tree: a dynamic index for multi-dimensional objects. In: Proceedings of the 13th Conference on Very Large Databases, pp. 507–518. Brighton, UK (1987)
39. Su, Z., Li, S., Zhang, H.: Extraction of feature subspace for content-based retrieval using relevance feedback. In: Proceedings of ACM Multimedia, Ottawa, Canada (2001)

40. Tong, S., Chan, E.: Support vector machine active learning for image retrieval. In: Proceedings of ACM Multimedia, Ottawa, Canada (2001)
41. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, Berlin Heidelberg New York (1995)
42. Vertan, C., Boujemaa, N.: Embedding fuzzy logic in content based image retrieval. In: Proceedings of the 19th International Meeting of the North America Fuzyy Information Processing Society, Atlanta (2000)
43. Wang, J.Z., Du, Y.P.: Scalable integrated region-based image retrieval using irm and statistical clustering. In: Proceedings of the ACM and IEEE Joint Conference on Digital Libraries, Roanoke, VA (2001)
44. Wang, J.Z., Li, J., Wiederhold, G.: Simplicity: semantics-sensitive integrated matching for picture libraries. IEEE Trans. Pattern Anal. Mach. Intell. **23**(9), 947–963 (2001)
45. White, D.A., Jain, R.: Similarity indexing with the *ss*-tree. In: Proceedings of the 12th IEEE International Conference on Data Engineering (1999)
46. Zhang, R., Zhang, Z.: Addressing cbir efficiency, effectiveness, and retrieval subjectivity simultaneously. In: 5th ACM International Workshop on Multimedia Information Retrieval, Berkeley, CA. In conjunction with ACM Multimedia (2003)
47. Zhang, R., Zhang, Z.: Hidden semantic concept discovery in region based image retrieval. In: IEEE International Conference on Computer Vision and Pattern Recogntion (CVPR), Washington, DC (2004)
48. Zhang, R., Zhang, Z.: Stretching bayesian learning in the relevance feedback of image retrieval. In: Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic (2004)
49. Zhang, R., Zhang, Z., Khanzode, S.: A data mining approach to modeling relationships among categories in image collection. In: 10th ACM International Conference on Knowledge Discovery and Data Mining, pp. 749–754, Seattle, WA (2004)