ACM Multimedia Systems Journal manuscript No.
(will be inserted by the editor)

Interactive Pattern Analysis for Relevance Feedback in Multimedia Information Retrieval

Yimin Wu[1], **Aidong Zhang**[2]

Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY 14260, USA; Email: {yiminkwu, azhang}@cse.buffalo.edu

July, 2004, Vol. 10, No.1 : 41-55

Abstract Relevance feedback is a mechanism to interactively learn the user's query concept online. It has been extensively used to improve the performance of multimedia information retrieval. In this paper, we present a novel interactive pattern analysis method, which reduces relevance feedback to a two-class classification problem and classifies multimedia objects as relevant or irrelevant. To perform interactive pattern analysis, we propose two online pattern classification methods termed interactive random forests (IRF) and adaptive random forests (ARF), which adapt a composite classifier known as random forests for relevance feedback. IRF improves the efficiency of regular random forests (RRF) with a novel two-level resampling technique termed biased random sample reduction, while ARF boosts the performance of RRF with two adaptive learning techniques called dynamic feature extraction and adaptive sample selection. During interactive multimedia retrieval, both ARF and IRF run 2-3 times faster than RRF, while achieving comparable precision and recall against the latter. Extensive experiments on a COREL image set (with 31,438 images) demonstrate that our methods (i.e., IRF and RRF) achieve at least a 20% improvement on average precision and recall over the state-of-the-art approaches.

Key words Multimedia information retrieval, relevance feedback, interactive pattern analysis, random forests.

1 Introduction

In multimedia databases, multimedia objects (such as images and video clips, etc.) are indexed by their content descriptors, which are usually real-valued feature vectors. To search multimedia databases, a query-by-example scheme is often used, where the query is represented by one or several example object(s). In response to the user's query, a retrieval system ranks database objects using some query mechanism. In the earlier years of content-based multimedia (and image) retrieval, most multimedia retrieval systems employ built-in query mechanisms, which always respond to a given query with the same retrieval results. For example, one of the extensively used query mechanism (by earlier systems) is to rank database objects using their distances to the query, where the distance (between a database object and the query) is calculated from their feature vectors using some fixed distance metric. The built-in query mechanisms often fail to achieve good performance in practice, because they are unable to address the following issues:

- The subjective nature of information retrieval. That is, different users have different interpretations or preferences toward the same multimedia object.
- Disparity between the machine and the user in representing multimedia information. In other words, content-based multimedia descriptors, i.e., feature vectors, are unable to capture the user's query concepts well.

A common solution for the above problems is to design a flexible query mechanism, which can adapt its retrieval results to the user's query concept(s) online. This is often achieved by an online learning technique termed as relevance feedback. During multimedia retrieval, relevance feedback allows the user to participate in retrieval loop as follows:

- The system obtains an initial retrieval result using the built-in query mechanism.
- It then asks the user to label all (or just some) returned objects as relevant or irrelevant.
- Using user-labeled objects as training examples, the system iteratively refines its query mechanism to capture the user's query preference(s) or concept(s).

As a critical component of modern multimedia databases, relevance feedback has been explored soundly in the past few years, and many promising methods have been proposed so far. Most existing approaches [25, 32] try to find an optimal query and feature weights, while some methods [14, 24] reduce relevance feedback to an optimization problem, which is solvable by a pre-whitening [11] transformation. More recently, many researchers address relevance feedback using some machine learning and/or pattern recognition methods. For instance, some researchers apply the Bayesian framework

[7,30,27], while others employ biased discriminant analysis [35], AdaBoost algorithm [28], or support vector machine [29].

1.1 Related Work

Recent research results [31] indicate that relevant objects tend to distribute multimodally, especially when the query concepts are complex (such as those represented by high-level semantics). However, many existing methods [24,28,32] assume a unimodal distribution for relevant objects, so they only perform optimally in unimodal cases.

To address the multimodality (of relevant objects), some researchers [18,29] reduce relevance feedback to a classification problem and address it using traditional classifiers such as decision tree [23] or support vector machine (SVM) [15]. Unfortunately, during interactive multimedia information retrieval, these traditional classification methods can not yield strong classifiers due to the following reasons:

- **Limited number of training examples**. With the user in the retrieval loop, we can only anticipate a very limited number of labeled training examples from the user.
- **High dimensionality of multimedia feature spaces**. Multimedia feature spaces, such as color histogram, can have hundreds of dimensions.

In this case, as pointed out by Breiman [3,4], a tree classifier can only achieve a classification accuracy slightly better than choosing classes by chance. Similarly, support vector machine can not yield a strong classifier in relevance feedback, unless more training examples can be obtained from the user (as in [29]).

Recent researches [3,10,4] in machine learning show that we can construct a strong (composite) classifier by combining multiple weak classifiers, each of which just slightly outperforms choosing classes by chance. Among the state-of-the-art approaches for growing composite classifiers, AdaBoost [10] has been applied to relevance feedback in image retrieval [28]. However, the AdaBoost-based approach (ADA) [28] only performs optimally in unimodal-Gaussian case, because it trains each classifier with only one feature [1]. On each feature, ADA trains a weak classifier using the two-class Gaussian model. That is, it computes a Gaussian model to fit relevant and irrelevant objects, respectively. When all the features are taken into account, ADA assumes a unimodal distribution for either class of objects.

To address multimodality, we develop two online and nonparametric pattern classification algorithms using random forests [4], which trains a composite classifier from multiple tree classifiers. As a combination of tree classifiers, random forests is a nonparametric and nonlinear classification algorithm with proven good performance on many traditional pattern classification problems.

[1] A feature denotes an element of the feature vector.

1.2 Our contribution

In this paper, we present an interactive pattern analysis method, which aims to improve the performance of multimedia retrieval using relevance feedback. During multimedia information retrieval, our method employs an online pattern analysis technique to learn the distribution pattern of relevant objects; and it intelligently reduces irrelevant objects from the database. From the reduced object set, our approach refines its previous retrieval results by returning the top-k nearest neighbors (of the query) to the user.

To facilitate interactive pattern analysis, we first present a novel two-level resampling technique called *biased random sample reduction* (BRSR), which aims to improve the efficiency of random forests by reducing the size of training set for each tree classifier. By combining random forests with BRSR, we develop the online pattern classification algorithm *interactive random forests (IRF)* for relevance feedback. Our extensive experimental results [33] have demonstrated the effectiveness and robustness of IRF for interactive multimedia retrieval.

Effective as it is, BRSR does not distinguish between the *most-informative* training examples (which comprise the relevant examples and the centroids of irrelevant ones) and the *less-informative* ones, because it just randomly selects irrelevant examples – in case the training set became oversized for online learning. To perform sample selection more effectively, we present an *adaptive sample selection* method to locate the *most-informative* examples in the training set. With *adaptive sample selection*, we improve the efficiency of interactive pattern analysis by using only the *most-informative* training examples for relevance feedback. To further boost the performance of our interactive pattern analysis, we propose the *dynamic feature extraction* approach, which aims to alleviate the curse of dimensionality by extracting an optimal subspace (of the original feature space) using balanced information gain [8]. In our experiments, the *dynamic feature extraction* can remove up to 70% of all features, while causing at most 3% degradation on average precision and recall. By combining random forests with the adaptive learning techniques, we present another online pattern classification algorithm called *adaptive random forests (ARF)*.

With the above two online pattern classification algorithms (i.e, IRF and ARF), we reduce relevance feedback to a two class classification problem and classify database objects using either IRF or ARF. Our experiments show that both IRF and ARF run 2 to 3 times faster than the regular random forests, while achieving comparable precision and recall against the latter. Extensive experiments on a large-scale image database (with 31,438 COREL images) demonstrate that our method outperforms the state-of-the-art approaches [15,28] by at least 20% on average precision and recall.

The rest of this paper is organized as follows. We first coin some useful notations and introduce the random forests algorithm in Section 2. Our interactive pattern analysis method is then presented in Section 3. Empirical results are provided in

Section 4. We give further discussions in Section 5 and conclude in Section 6.

2 Random Forests

In this section, we first give some notation and concepts that will be used in the rest of the paper. We then introduce the pattern classification algorithm random forests [4].

2.1 Notation and Concepts

As usual, we represent the feature space as $F = \{f_1, ..., f_M\}$, where M is the dimensions of F. We denote the multimedia database by $db = \{o_1, ..., o_I\}$, where I is the size of db. To represent each multimedia object $o_i \in db$, we use a real-valued vector (i.e., point) $\mathbf{o_i} = [o_{i,1}, ..., o_{i,M}]^T$, where $\mathbf{o_i}$ is an instance in the feature space F. Similarly, the query q is represented by vector $\mathbf{q} = [q_1, ..., q_M]^T$. To calculate the distance between object o_i and query q, we use the Euclidean distance of their feature vectors.

With the above notation, we can now elaborate some useful definitions for our *dynamic feature extraction*. From the original feature space F, our *dynamic feature extraction* aims to extract an optimal M^* dimensional subspace $F^* \subset F$ and a projection $\psi : F \mapsto F^*$. With projection ψ, we can then project the object $\mathbf{o_i} \in db$ into F^*, with $\mathbf{o_i^*} = \psi(\mathbf{o_i})$ and $\psi(\mathbf{o_i}) \in F^*$.

We now specify the notation for *online pattern classification*. During interactive multimedia retrieval, we obtain the training sample $S = \{(s_1, v_1)..., (s_N, v_N)\}$ from relevance feedback, where N is the size of S. In the training set S, each training example $(s_n, v_n) \in S$ is a labeled object represented by $\mathbf{s_n} = [s_{n,1}, ..., s_{n,M}]^T$, where $v_n \in \{0, 1\}$ is its class value (0/1 means irrelevant/relevant). We then denote relevant/irrelevant training sample as R/U, where $S = R \bigcup U$. With training sample S, we train an online random forest h^* (using either IRF or ARF) to classify database objects into one of the two classes: relevant or irrelevant.

2.2 Random Forests Algorithm

Random forests [4] is a method for growing a composite classifier from multiple randomized tree classifiers. To obtain the composite classifier, it combines the bagging [3] technique with random feature selection [4] and achieves favorable performance over the state-of-the-art approaches (such as bagging and AdaBoost [10]).

The tree classifier trained in random forests is the classification and regression tree (CART) [5]. At each node, CART searches through all features to find the best split for allocating training data to two children nodes. If the current node only contains training data of one specific class, CART makes it a leaf node. This leaf node then assigns all test data falling into it to that class. All test instances are run down the tree to get their predicted classes. Since a feature can be used to split multiple times, the CART is able to handle multimodal distribution of relevant points.

A random forest $h = \{h_j(\mathbf{o}, \theta_j), j = 1, ...J\}$ is a collection of J randomized CARTs. In random forest h, the jth CART h_j is trained with a bootstrap [3,9] sample $S_j \subset S$ and a random vector θ_j. The bootstrap sample S_j is obtained by randomly resampling the original training set S with replacement from S, while the random vector θ_j is generated independent of the past random vectors θ_1,..., θ_{j-1}, but with the same distribution. Although the random vectors in random forests may seem complicated, they serve as nothing more than a formal definition of the random feature selection technique [4]. That is, when growing a CART $h_j \in h$, instead of searching all the M features (in the feature space F) to get each split, we only examine $M' < M$ randomly selected features every time. As pointed out by Breiman [4], a random forest is insensitive to M' and performs optimally (in general) when $M' = \sqrt{M}$. So, a random forest runs $\sqrt{M}$ times faster than bagging and AdaBoost for combining tree classifiers.

To classify input object $\mathbf{o}$, random forest h lets its member classifiers vote for the most popular class, where each classifier casts a unit vote. Breiman [4] proved that a random forest, as more trees are added, does not overfit and only produces a limited value of generalization errors. In comparison with AdaBoost, random forests performs as good as AdaBoost, but sometimes better. Moreover, random forests is more robust than AdaBoost on noisy data.

3 Interactive Pattern Analysis

In this section, we present our interactive pattern analysis method, which consists of two online pattern classification algorithms (i.e., interactive random forests and adaptive random forests) and their major components: biased random sample reduction, dynamic feature extraction, and adaptive sample selection.

3.1 Interactive Random Forests

During interactive multimedia retrieval, the computational cost of training a regular random forest can become prohibitively high. To address this issue, we propose the *interactive random forests*, which adapts random forests for online learning.

3.1.1 Biased Random Sample Reduction To improve the efficiency of relevance feedback, our interactive random forests employs the two-level resampling technique *biased random sample reduction* (BRSR) to reduce the size of training set for each tree classifier. Before introducing BRSR, we first present an analysis of the computational complexity (of regular random forests), which is dominated by training the tree classifiers (CART). If each CART is grown to a uniform depth D (i.e., to 2^D leaf nodes), the complexity $T(h)$ of random forest h is given by [4]:

$$T(h) \propto J * N' * (D + 1) * \log N', \qquad (1)$$

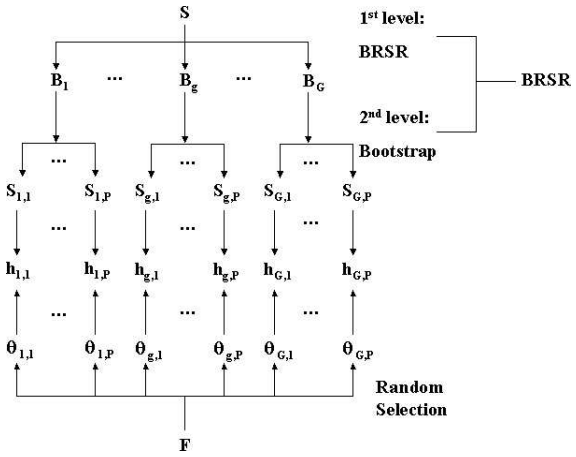


Fig. 1 Interactive random forests.

where J is the number of tree classifiers and N' is the size of each bootstrap sample. Since $T(h)$ increases super-linearly with N', we can reduce $T(h)$ by ensuring $N' \leq N_0$, where N_0 is a threshold. However, in the regular random forests, N' always (approximately) equals $0.63N$, because the size of a bootstrap sample is about 63% of that of the original training sample [9].

For online learning, bootstrap lacks the flexibility to reduce the size of each bootstrap sample as necessary. To address this issue, we employ a novel two-level resampling approach termed *biased random sample reduction (BRSR)* to ensure $N' \leq N_0$. Figure 1 shows how BRSR works in interactive random forests (IRF). To reduce the sample size for each tree classifier, our method employs BRSR to generate a *reduced sample set* $B = \{B_g | g = 1, ..., G, \ and \ B_g \subset S\}$, where $G = |B|$. From each reduced sample $B_g \in B$, we create $P = J/G$ [2] bootstrap samples $S_g = \{S_{g,p}, p = 1, ..., P\}$, with the restriction that $|S_{g,p}| \leq N_0$ (for $\forall\ g \in [1, G]$ and $p \in [1, P]$). As shown in Figure 1, we use BRSR to denote the two-level resampling technique as a whole since the first level (i.e., BRSR) is the one performing online sample reduction. From the resulting samples of our two-level resampling approach, our method trains a tree classifier $h_{g,p}$ with each sample $S_{g,p}$ and a random vector $\theta_{g,p}$. Hence, our interactive random forests $h^i = \{h_{g,p}, g = 1, ..., G, p = 1, ..., P\}$ contains $J = G * P$ tree classifiers, and it allows all the member classifiers to vote when classifying input data points.

To present more details about BRSR, we need to coin several useful definitions. At first, we define the random sample reduction (RSR) approach as follows:

Definition 1 *Random sample reduction (RSR) generates, from the given training sample S, G reduced samples $B = \{B_g | g = 1, ..., G, \ and \ B_g \subset S\}$, with $\bigcup_{g=1}^{G} B_g = S$; $B_g \cap B_x = \phi, \ for \ \forall g \neq x$; and $|B_g| = |S|/G$. To generate B, we can randomly permute S, and divide the permuted examples into G non-overlapped and equally sized subsets.*

[2] Without loss of generality, we assume the tree number J is always divisible by the number of reduced samples G.

However, during multimedia retrieval, the number of relevant training examples is often much smaller than that of irrelevant ones, especially after multiple iterations are run. So, it is often favorable to reduce more irrelevant examples than relevant ones. This leads to the following definitions:

Definition 2 *Relevant-bootstrap irrelevant-RSR (RBIRSR) generates the reduced sample set as $B = \{B_g = B_g^R \cup B_g^U, g = 1, ..., G\}$, where B_g^R is a bootstrap sample of the relevant sample R, and B_g^U is the gth RSR sample generated from irrelevant sample U. The size of each reduced sample can then be calculated by*

$$\alpha = 0.63 * |R| + |U|/G. \tag{2}$$

We call α the RBIRSR sample size.

Based on the above definitions, we define our biased random sample reduction (BRSR) as follows:

Definition 3 *Biased random sample reduction employs RBIRSR to generate G reduced samples in B if the RBIRSR sample size $\alpha \leq 1.5 * N_0$. Otherwise, it uses RSR to generate the G reduced samples.*

Whenever possible, BRSR reduces more irrelevant examples than it does to relevant ones. This is why it is named a *biased sample reduction* method.

Condition 1	Condition 2	G	Sampling Method
$N \leq 1.5N_0$	true	1	Bootstrap
$(G_0-1)*N_0 < N \leq G_0 * N_0$	$\alpha \leq 1.5N_0$	G_0	RBIRSR
$(G_0-1)*N_0 < N \leq G_0 * N_0$	$\alpha > 1.5N_0$	G_0	RSR

Table 1 An illustration of our sample reduction method, where $G_0 = \lceil N/N_0 \rceil$ and $G_0 \geq 2$.

Table 1 demonstrates how to calculate the value of G and how to choose the method to obtain reduced samples. When both conditions 1 and 2 are satisfied, G is set to the value given in the third column, and the reduced samples are generated using the method specified in the fourth column. Table 1 illustrates the following three cases of our method:

- If $N \leq 1.5N_0$, BRSR is equivalent to one-level bootstrap, so the size of each training set is $0.63 * 1.5 * N_0 < N_0$ (the size of a bootstrap sample is 63% of the original sample [9]).
- If $N > 1.5N_0$ and $\alpha \leq 1.5N_0$ (cf. Formula 2), we use RBIRSR on the first level. Hence, the size of each reduced sample $B_g \in B$ (see Definition 2) equals α; and the size of each bootstrap sample $S_{g,p}$ (cf. Figure 1) is $0.63 * \alpha \leq 0.63 * 1.5 * N_0 < N_0$.
- If $N > 1.5N_0$ and $\alpha > 1.5N_0$, we employ RSR (see Definition 1) to generate $G = \lceil N/N_0 \rceil$ reduced samples on the first level. Since the size of each reduced sample B_g is $N/G \leq N_0$, the size of each bootstrap sample $S_{g,p}$ will be $0.63N/G < N_0$.

From the above discussions, each bootstrap sample $S_{g,p}$ is no larger than N_0 in all cases.

Algorithm 1 *Interactive Random Forests*

Input 1) training set $S = \{(s_1, v_1)..., (s_N, v_N)\}$, where $v_n = 1$ or 0; 2) feature space $F = \{f_1, ..., f_M\}$; 3) J: the number of CARTs to grow; 4) N_0: threshold of sample size.
Initialize: set the interactive random forest $h^i \Leftarrow \{\}$.
Generate reduced sample set $B = \{B_g | g = 1, ..., G$, and $B_g \subset S\}$ using BRSR;
for $g = 1$ to G **do**
  $P \Leftarrow J/G$;
  for $p = 1$ to P **do**
    Generate a bootstrap sample $G_{g,p}$ from $B_g \in B$;
    Train a randomized CART $h_{g,p}$ with $G_{g,p}$;
    $h^i \Leftarrow h^i \cup \{h_{g,p}\}$;
  end for
end for
Output: the interactive random forests:

$$h^i(\mathbf{o}) = \begin{cases} 1 \text{ if } \sum_{g=1}^{G} \sum_{p=1}^{P} h_{g,p}(\mathbf{o}) \geq \frac{J}{2}, \\ 0 \text{ otherwise.} \end{cases} \quad (3)$$

3.1.2 Interactive Random Forests We combine random forests with BRSR and present *interactive random forests (IRF)* in Algorithm 1. According to the size of training sample S, our IRF has to adjust several important parameters (such as the method for generating reduced samples and the number of reduced samples G, etc.). Since the training sample S is accumulated from the *interaction* between the user and the machine, our IRF *interactively* adapts its way to train the member tree classifiers. Hence, we term Algorithm 1 as an *interactive* one.

By employing BRSR, our IRF accommodates the capability to adjust the computational cost according to the system performance. As to the effectiveness, BRSR may have double-bladed effect on the classification accuracy. On the one hand, it may harm the classification accuracy by worsening the overfitting problem, since less trees are trained from each reduced sample B_g [4]. On the other hand, it can improve the performance by reducing the correlation of generalization errors among tree classifiers, which usually improves classification accuracy of the random forest [4]. Our extensive experiments [33] indicated that IRF can improve the efficiency by $2-3$ times, while only causing trivial degradation on search quality.

3.2 Adaptive Random Forests

To further boost the effectiveness of our interactive pattern analysis, we present two adaptive learning techniques termed as *dynamic feature extraction* and *adaptive sample selection*. Dynamic feature extraction aims to alleviate the curse of dimensionality by extracting an optimal subspace (of the original feature space) using balanced information gain [8], while adaptive sample selection improves the efficiency of pattern analysis by choosing the most-informative examples for relevance feedback. With the adaptive learning techniques, we develop another online pattern classification algorithm called *adaptive random forests*.

3.2.1 Dynamic Feature Extraction The high dimensionality of multimedia feature spaces often causes the *curse of dimensionality* [11] in relevance feedback. That is, it not only impedes the application of many pattern classification methods, but also degrades the efficiency of the learning machine. To alleviate the *curse of dimensionality*, some researchers extract a low-dimensional subspace using PCA (principle component analysis) [27], while others [28] employ the feature selection technique, which aims to select optimal features from given feature space according to some optimization criteria.

Although feature selection has been studied for several decades, most existing methods are too computationally intensive for online learning. To address this issue, we present an efficient online feature selection method termed *dynamic feature extraction*, whose complexity is only linearly related to M. To extract the optimal features, our approach employs *balanced information gain* [8], which was originated from information theory and machine learning. To present our method, we first define the *entropy* [19] of training set S as follows:

$$E(S) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}, \quad (4)$$

where $p_{\oplus}/p_{\ominus}$ is the percentage of relevant/irrelevant training examples in S. The entropy E(S) indicates the *purity* of training set S: $E(S) = 0$ if S contains only relevant or irrelevant examples; $E(S) = 1$ if S contains equal number of relevant and irrelevant examples. According to information theory, the larger the entropy $E(S)$, the more number of bits is required to encode S. Let $\biguplus S_i$ be a partition of S, the *information gain* [19] of partition $\biguplus S_i$ is then given by:

$$G(\biguplus_i S_i) = E(S) - \frac{1}{|S|} \sum_i |S_i| E(S_i). \quad (5)$$

In the light of Occam's razor [19], the partition maximizing the above information gain should be chosen, since it leads to the most concise representation of S. Despite its successful applications in text classification [34], information gain has the drawback of not placing any penalty on the arity of partitions, so it favors partitions with excessively large arity [16]. To balance the bias of information gain, we employ the following *balanced information gain* [8]:

$$B_g(\biguplus_i S_i) = \frac{G(\biguplus_i S_i)}{\log_2 \kappa}, \quad (6)$$

where κ is the arity of $\biguplus S_i$. Elomaa *et al.* [8] demonstrated the effectiveness of the above *balanced information gain* in training multisplitting tree classifiers, and we will present how to use it to extract the optimal features, which well facilitate our interactive pattern analysis.

So far we have reduced the feature extraction problem to selecting features with maximum *balanced information gain*. Figure 2 presents some useful concepts for calculating (balanced) information gain. To calculate the information gain of a feature, we first sort all examples in ascending order, and set the mean values of adjacent examples with different classification as potential *cut points* [19,8] of the partition. On each feature, t-1 *cut points* create t continuous *intervals*, which

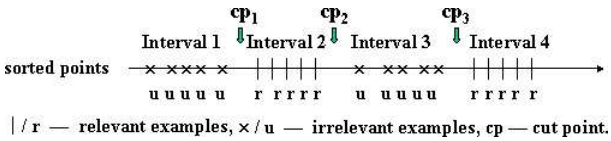


Fig. 2 Several useful concepts for computing (balanced) information gain.

comprise a t-ary partition of S, since these intervals contain t non-overlapped subsets of S.

Algorithm 2 *Dynamic Feature Extraction*

Input 1) training set $S = \{(s_1, v_1)..., (s_N, v_N)\}$, where $v_n = 1$ or 0; 2) feature space $F = \{f_1, ..., f_M\}$; 3) M^*: dimensions of the subspace F^*.
Initialize: set the subspace $F^* \Leftarrow \{\}$.
for features from f_1 to f_M **do**
  Sort training examples in ascending order on the current feature;
  Set the mean values of adjacent examples with different classification as potential *cut points* for the partition;
  Choose (from all possible 2-ary partitions) the optimal 2-ary partition as the one with maximum *information gain*.
  Obtain the κ-ary partition by partitioning one of the existing κ - 1 partitions to a 2-ary partition.
  Calculate the *balanced information gain* of current feature as the best value achieved by the partitions from 2 to κ^*-ary, where κ^* is the maximum arities desired;
end for
Obtain M^* features with the largest balanced information and add them into F^*.
Output: the subspace F^* and the projection $\psi : F \mapsto F^*$.

Based on the above discussion, we present our *dynamic feature extraction* in Algorithm 2. Denote the coordinates of the M^* optimal features (extracted by Algorithm 2) by the sequence $Z = \{1 \leq z_1 < ... < z_m < ... < z_{M^*} \leq M\}$. For any point $\mathbf{o_i} \in F$, its projection $\mathbf{o_i^*} = \psi(\mathbf{o_i}) \in F^*$ is obtained by sequentially taking the elements (of $\mathbf{o_i}$) whose coordinates appear in Z. The computational complexity $T(e)$ of our feature extraction method is dominated by the sort operation, so we have $T(e) = O(MN \log N)$.

3.2.2 Adaptive Sample Selection To improve the efficiency of relevance feedback, our interactive random forests algorithm employs the biased random sample reduction (BRSR) to reduce the size of training sample for each tree classifier. Effective as it is, BRSR does not distinguish between the *most-informative* training examples and the *less-informative* ones, because it just randomly selects irrelevant examples (in case the training sample became oversized for online learning). To perform sample selection more effectively, we present an *adaptive sample selection* method to select no more than N_0 *most-informative* training examples for each tree classifier. In the training set S, the *most-informative* examples are given as follows

– **Relevant examples**. In relevance feedback, the number of relevant examples is often much smaller than that of irrelevant ones. So, every relevant example is *precious* and *informative*.
– **Centroids of irrelevant examples**. According to pattern recognition [11], the centroids of irrelevant clusters are dependable representatives for irrelevant examples/patterns.

The above-mentioned examples provide more information about the distribution patterns of multimedia objects, so they are more *informative* than other training examples. To find centroids of irrelevant examples, we cluster irrelevant training set with an incremental clustering method termed doubling algorithm [6], which provides proven performance guarantee on the quality of the resulting clusters. Doubling algorithm takes the maximum number of clusters ζ as input. When a new point $\mathbf{o}$ is added, the algorithm either merges it into some existing cluster K or creates a new singleton cluster with $\mathbf{o}$, depending on the current value of the radius. To make sure the number of clusters $\nu \leq \zeta$, the clustering algorithm doubles the radius and merges existing clusters whenever $\nu > \zeta$. If two clusters are merged, the algorithm sets the centroid of the resulting cluster to either centroid of the merged clusters. Moreover, when point $\mathbf{o}$ is merged into cluster K, the centroid of the resulting cluster is set to that of K. From the above discussions, we can see that doubling algorithm is a centroid-preserving approach, because it guarantees that the centroids of resulting clusters are from the clustered data points. Porkaew *et al.* [22] employed this algorithm to cluster relevant images in their query expansion scheme. Different from their purpose, our goal is to find the most-informative irrelevant examples, so we cluster irrelevant sample using doubling algorithm.

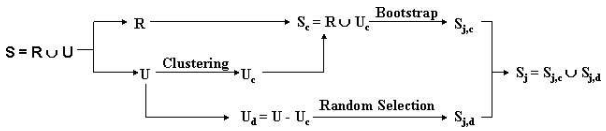


Fig. 3 An illustration of our adaptive sample selection.

Figure 3 demonstrates the principles of our adaptive sample selection method, which is summarized as follows:

– First, incrementally cluster new irrelevant examples in each feedback iteration and add the resulting centroids into the *most-informative* irrelevant sample U_c.
– Second, get the *most-informative* sample $S_c = R \bigcup U_c$, where R is the relevant training sample.
– Third, get the *less-informative* sample $U_d = U - U_c$.

To obtain the training sample S_j (for the jth classifier h_j), we

– create a bootstrap sample $S_{j,c}$ from S_c, and
– randomly choose $N_0 - |S_{j,c}|$ examples from U_d to form the *random sample* $S_{j,d}$.

Finally, we use $S_j = S_{j,c} \bigcup S_{j,d}$ as the training sample for h_j. In comparison with bootstrap, our adaptive sample selec-

tion distinguishes between *most-informative* training examples and *less-informative* ones. To reduce the number of training examples, our method selects as many *most-informative* training examples as bootstrap does, but intelligently discards some *less-informative* ones.

The computational complexity $T(s)$ of our adaptive sample selection is dominated by the clustering operation. And hence, we have $T(s) = O(N_1 \zeta \log \zeta)$ [6], where N_1 is the number of new irrelevant examples in each iteration and ζ is the maximum number of clusters.

Algorithm 3 *Adaptive Random Forests*

1: **Input** 1) training set $S = \{(s_1, v_1)..., (s_N, v_N)\}$, where $v_n = 1$ or 0; 2) feature space $F = \{f_1, ..., f_M\}$; 3) J: the number of CARTs to grow; 4) N_0: threshold of sample size.
2: **Initialize:** set the adaptive random forest $h^a \Leftarrow \{\}$.
3: **Call** Algorithm 2 to learn the subspace F^* and the projection $\psi : F \mapsto F^*$;
4: **Project** all the training examples into the subspace F^* and denote the set of projected samples as S^*.
5: **for** $j = 1$ to J **do**
6: **Generate** the training set S_j^* using the *adaptive sample selection* approach;
7: **Train** a randomized CART h_j with S_j^*;
8: **Set** $h^a \Leftarrow h^a \bigcup \{h_j\}$;
9: **end for**
10: **Output**: the subspace F^*, the projection $\psi : F \mapsto F^*$ and the adaptive random forests:

$$h^a(\mathbf{o}) = \begin{cases} 1 \text{ if } \sum_{j=1}^{J} h_j(\psi(\mathbf{o})) \geq \frac{J}{2}, \\ 0 \text{ otherwise.} \end{cases} \quad (7)$$

3.2.3 Adaptive Random Forests With the adaptive learning techniques, i.e, dynamic feature extraction and adaptive sample selection, we present the online pattern classification algorithm *adaptive random forests* in Algorithm 3. The adaptive random forests algorithm is superior to interactive random forests in the following two aspects:

– By using adaptive sample selection, it performs the sample selection more effectively by distinguishing between the *most-informative* training examples and the *less informative* ones.
– With dynamic feature extraction, it not only further boosts the efficiency of relevance feedback by removing the redundancy in the feature space, but also accommodates the ability to overcome the curse of dimensionality.

Our experiments (see Section 4) indicate that ARF consistently outperforms IRF. ARF improves the average precision and recall over IRF by $1 - 2\%$. Moreover, it also runs about 20% more efficiently than the latter in most cases.

3.3 Interactive Pattern Analysis

During multimedia information retrieval, we train an online random forest h^* (using either IRF or ARF) to classify database objects. We use classified-relevant/irrelevant to denote objects (or set of objects) that are classified as relevant/irrelevant. From the classified-relevant objects, we return the k nearest neighbors of query q to the user. However, the online random forest occasionally outputs less than k classified-relevant objects. To address this issue, we define the *relevance probability* of object o as follows:

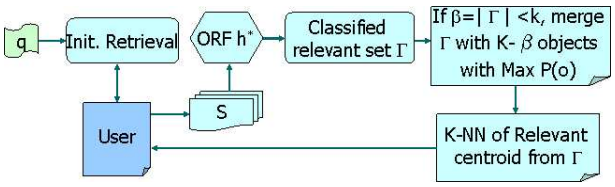


Fig. 4 The fbwchart of interactive pattern analysis, where ORF denotes online random forests trained with either IRF or ARF.

Definition 4 *The relevance probability $P(1|\mathbf{o})$ of object o is the the number of classifiers that classify it as relevant over the total number of classifiers. Its formal definition is given by*

$$P(1|\mathbf{o}) = \begin{cases} \frac{\sum_{g=1}^{G} \sum_{p=1}^{P} h_{g,p}(\mathbf{o})}{J} & \text{if } h^* = h^i, \\ \frac{\sum_{j=1}^{J} h_j(\psi(\mathbf{o}))}{J} & \text{if } h^* = h^a, \end{cases} \quad (8)$$

where $h^* = h^i$ (or h^a) means h^* is trained with IRF (or ARF).

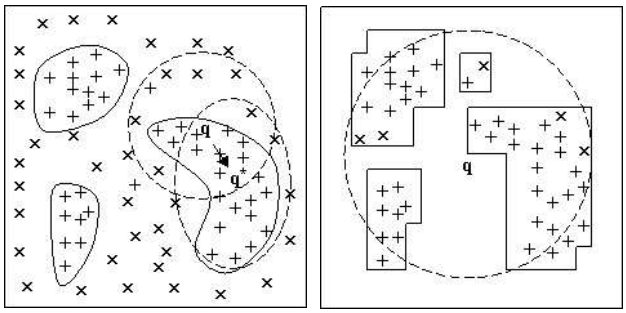
The larger the $P(1|\mathbf{o})$, the more confident it is to output object o as relevant. So, our method returns some classified-irrelevant objects with the largest $P(1|\mathbf{o})$, in case less than k objects were classified as relevant.

Algorithm 4 *Interactive Pattern Analysis*

Input 1) training set $S = \{(s_1, v_1)..., (s_N, v_N)\}$, where $v_n = 1$ or 0; 2) q: the query point; 3) the feature space: $F = \{f_1, ..., f_M\}$; 4) J: the number of CARTs to grow; 4) N_0: threshold for the size of training sample; 5) W: the maximum iteration number; 6) k: the number of nearest neighbors returned to the user.

1: **for** $w = 1$ to W **do**
2: **Call** Algorithm 1 or Algorithm 3 to train an online random forest h^*.
3: **Classify** all objects to get the classifi ed-relevant set Γ.
4: **if** $|\Gamma| < k$ **then**
5: **Add** $k - |\Gamma|$ classifi ed-irrelevant objects (with the largest relevance probability) into Γ.
6: **end if**
7: **Find** the k nearest neighbors of q from Γ, and return them to the user.
8: **Obtain** the new training examples from the user, and merge them with S.
9: **Update** query q by setting it to the centroid of all relevant objects, that is, set the the feature vector of q to the average value of all relevant feature vectors.
10: **end for**

For each query q, our method runs an initial database search and returns the k nearest neighbors of q to the user.



(a) Feature reweighting approaches; (b) Our method.

Fig. 5 A comparison of traditional relevance feedback approaches and our method, where plus sign denotes relevant points and times sign denotes irrelevant ones.

It then asks the user to provide the initial training sample S by labeling retrieved objects as relevant or irrelevant. With S, our method then runs the *interactive pattern analysis* algorithm (presented in Algorithm 4) to improve the initial search results. Figure 4 demonstrates the flowchart of our interactive pattern analysis algorithm.

Figure 5 compares typical query results of different methods. Figure 5 (a) shows that the Euclidean-distance-based nearest neighbor search generates a circled boundary in the feature space. Its performance is usually poor, unless the query is located near the center of a large relevant cluster [3]. By using query movement and feature reweighting techniques, most traditional relevance feedback approaches [25, 24] can move the query toward the centroid of a relevant cluster. They also generate an elliptic hyper-surface to cover relevant points better. However, the common drawback of these methods is that they are prone to be trapped by local optimum, because they try to fit all relevant points with a Gaussian model. As a result, these approaches often try to adjust their queries to cover one relevant cluster as perfect as possible. Figure 5 (b) illustrates that our method, by applying interactive pattern analysis, can effectively overcome the multimodal distribution of relevant points, so it is able to locate relevant points from multiple disjoint relevant clusters.

4 Empirical Results

In this section, we evaluate the performance of our interactive pattern analysis method with extensive experiments on a real-world image database.

4.1 Experimental Setup

To test the performance of our method, we use a large-scale database with 31,438 Corel images, which are classified into hundreds of semantic categories (by Corel company). However, many of these categories (such as those titled by city names) are inappropriate for performance evaluation. So, we choose 44 semantic categories (such as rose, tiger, eagle, penguin and falls, etc.) with refined ground truth. We employ all the 5,172 images from these categories as queries. For each query, the retrieval is executed on the whole database (with 31,438 images). *Precision* and *recall* are used to evaluate the retrieval performance. *Precision* is the number of retrieved relevant images over the total number of retrieved images, and *recall* is the number of retrieved relevant images over the total number of relevant images in the database. To calculate precision and recall, only images from the same semantic category are considered as relevant. The average *precision* and *recall* of all queries are used as the overall performance.

Previous experimental results [17] indicated that better retrieval performance can be achieved by using a combination of various low-level image features (such as color, texture and shape, etc.). We performed extensive experiments to compare the retrieval performance achieved by diverse combinations of 10 low-level image features, such as: color moments [17], color histogram [20], color coherence vector [21] and Fourier shape descriptor [2]. The following is the combination of features which achieves the best retrieval performance:

- **Color**. We employ two color features in our experiments. The first one is a 64-bin color coherence vector [21] in the HSV color space. The second one is a 9-bin color moments [17] extracted from the L*a*b color space.
- **Texture**. We represent texture using a 10-bin wavelet-based feature [26].
- **Shape**. Shape information of an image can be well captured by its edge map. We obtain two statistical shape descriptors from the color edge map [2] of an image. The first one is the 64-bin edge coherence histogram [2], which represents the coherence information of edge pixels in the primary 8 directions. The second one is a 32-bin Fourier shape descriptor[2]. To get the Fourier descriptor, an image is first converted to the scale of 128×128 for edge detection. Afterwards, the 2D Fast Fourier Transformation is applied to the edge map. The magnitude image of the Fourier spectrum is then decimated by the factor of 16 to get the 32-bin Fourier shape descriptor.

We normalize feature values into the range [-1, 1] and concatenate all the 5 image features into a 179-bin feature vector.

The number of nearest neighbors returned (that is, k) is often called *scope*. Since retrieval performance also varies with scope, we conducted experiments on scopes of 20, 40, 80 and 120, respectively. For a comparison, we provide the performance of the AdaBoost-based method (ADA) [28] and support vector machine (SVM) [15] under the same experimental conditions. To train ADA, we combine 20 Gaussian-model-based weak classifiers as suggested in [28]; to train SVM, we employ the *SVM Light* [15] package and set all the parameters to their default values.

[3] Guo *et al. [12] demonstrated the defi ciency of nearest neighbor search using a fi gure similar to Figure 5 (a), but they still assumed a unimodal distribution for relevant objects.*

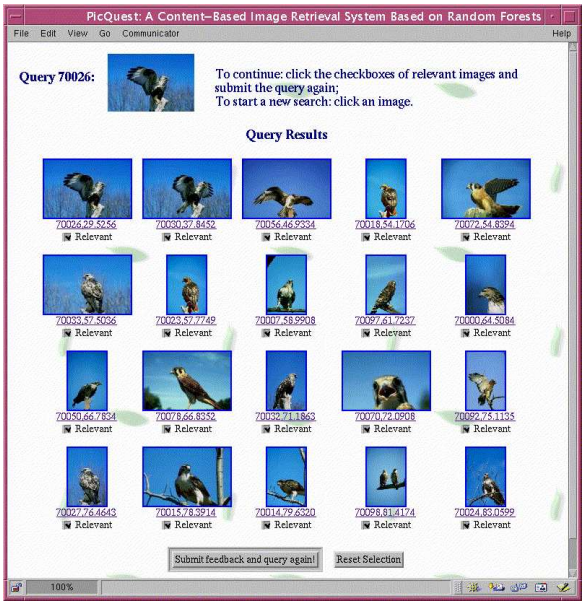


Fig. 6 User interface of our system.

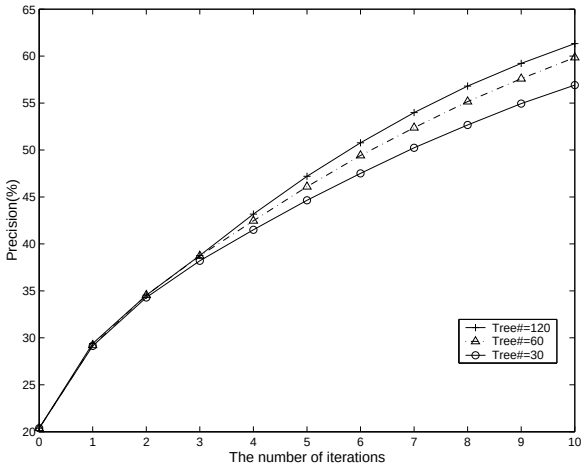


Fig. 7 Precisions achieved by 30, 60 and 120 trees.

N_0		IRF	ARF-179D (M^*=179)	ARF-120D (M^*=120)	ARF-60D (M^*=60)
Scope	20	90	90	90	120
	40	180	180	180	240
	80,120	240	180	180	240

Table 2 Value of N_0 in difference cases.

4.2 User Interface

Based on the presented method, we have implemented a content-based image retrieval system called *PicQuest*. Figure 6 demonstrates the user interface of our system. The current query is shown on the left top corner of the screen. The returned images are listed under the query sequentially. Under each returned image, two numbers, which are separated by a comma, are displayed. The first one is the unique number of the image. The second one is the distance between the image and the query. In the beginning, each image is assumed to be relevant. The user then identifies irrelevant images by disabling their *relevant* checkboxes. Afterwards, he or she can activate the *Submit* button to submit the relevance feedback. The user may cancel previous selections by enabling the *Reset Selection* button. He or she can start a new query at any time by clicking on a returned image or the *Go back to menu* option.

4.3 Thresholds

We demonstrate how to choose appropriate values for tree number J and sample size N_0 in this section. Figure 7 demonstrates the precision achieved by random forests on the scope of 20 when J=30, 60 and 120. We can see that 30 trees severely degrade the precision by 5% against 120 trees, so we need to train a random forest with at least 60 trees. On the other hand, the computational cost of the random forest increases linearly with the tree number J. Although 120 trees slightly outperform 60 trees by 1.5%, it is more cost-effective to train 60 trees, which achieve nearly optimal performance, while halving the computational cost of the random forest. Hence, we set J to 60 in all the following experiments.

To test the performance of our *dynamic feature extraction* method, we perform experiments when the dimension of the subspace (i.e., M^*) equals 179, 120 and 60, which correspond to employ all features, 60% of all features and 30% of them. To distinguish these three cases, we denote them by ARF-179D, ARF-120D and ARF-60D, respectively. For a specific value of M^*, we adjust N_0 to guarantee our adaptive random forests (ARF) runs as efficiently as the state-of-the-art approaches (such as ADA [28]). To achieve this goal, N_0 has to be no larger than 180 for ARF-179D (see Section 4.5). As for ARF-60D and ARF-120D, N_0 must be no larger than $\sqrt{179} * 180/\sqrt{M^*}$, since the computational cost of random forests is approximately proportional to $\sqrt{M^*}N_0$. However, the number of training examples never exceeds 180 on the scope 20. So, we set N_0 to 90 to demonstrate the effectiveness of our *adaptive sample selection* for this case.

For IRF to run as efficiently as ADA, N_0 should be no larger than 240. And hence, it is set to 240 on the scopes of 80 and 120 for IRF. Similar to ARF, N_0 is set to smaller values on the scopes of 20 and 40, since the number of training examples seldom grows larger than 240 on these scopes. Table 2 summarizes the above discussion and demonstrates the values of N_0 in all cases.

4.4 Distance Metrics

In our interactive pattern analysis method, we use unweighted Euclidean distance to calculate the distance between an image and the query. In this section, we compare the retrieval performance of unweighted Euclidean distance against a feature re-weighting method [32]. For each of the five image features (see Section 4.1), this feature re-weighting approach employs both relevant examples and irrelevant ones to learn

a weighted distance metric. It then calculates the overall distance (between an image and the query) as a weighted sum of the distances between all the five image features. Our previous experiments [32] demonstrated the effectiveness and robustness of the feature re-weighting method.

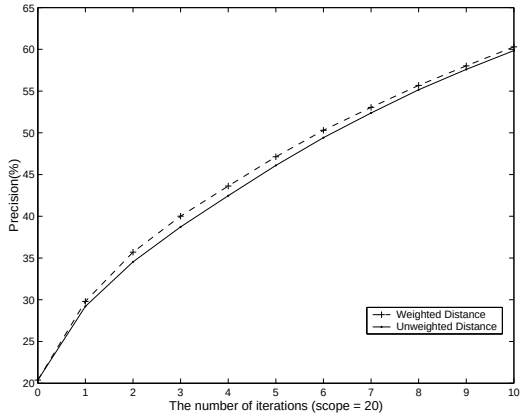


Fig. 8 Performance of weighted and unweighted Euclidean distances.

In this experiment, we first train a random forest with 60 trees to classify database objects. From the classified-relevant set, we then use either of the compared distance metrics to find the nearest neighbors of the query. Figure 8 compares the performance achieved by weighted and unweighted Euclidean distance on the scope of 20. We can see from Figure 8 that the weighted Euclidean distance does not achieve noticeable gain in performance over the unweighted one. The gain in precision (of weighted distance over unweighted one) is about 0.9% after the first iteration. It keeps decreasing as more iterations are run and is about 0.4% after 10 iterations. This experiment confirms our previous discussion (see Figure 5) that feature re-weighting techniques are prone to be trapped by local optimum, since they try to fit all relevant objects with a Gaussian model. When multiple clusters are involved, there is neither theoretic bases nor empirical evidences for the preference of weighted Euclidean distance to its unweighted counterpart. Hence, we use unweighted Euclidean distance in the remaining experiments.

4.5 Performance Evaluation

In the following experimental results, we do not present the recall on the scopes of 20 and 40, because (on smaller scopes) recall is not as effective as precision for comparing different approaches. For instance, in our experiments, the maximum achievable recall on the scope of 20 is about $16.7\%(=20/120)$, since the average size of all semantic categories is about 120. In this case, the difference on recall of compared methods does not precisely reflect their disparity in retrieval performance.

In our experiments, we run 10 feedback iterations for each query. Table 3 compares the performance of all online random forests (i.e., ARF-179D, ARF-120D, ARF-60D and IRF) for the 10th iteration. Among the 4 compared online random forests, ARF-60D is the most cost-effective one:

- It achieves the best retrieval performance on all scopes. On the scopes of 20, 80 and 120, ARF-60D improves the average precision and recall over all other online random forests by $1-3\%$. On the scope of 40, it achieves as good precision as other methods do.
- It runs about $15-20\%$ faster than ARF-120D and ARF-179D in most cases. In addition, ARF-60D improves the efficiency by about 20% over IRF on the scopes of 80 and 120.

Although ARF-60D runs about 20% slower than IRF on the scopes of 20 and 40, it is still the best one among all four compared online random forests. However, when efficiency is the key issue, IRF is still a good choice for smaller scopes, because the two-level resampling technique BRSR is less computationally-intensive than the dynamic feature extraction. On smaller scopes, the gain in efficiency achieved by dynamic feature extraction can not compensate its computational cost. Only when more training examples are involved on the scopes of 80 and 120, dynamic feature extraction manages to boost the efficiency (of ARF) noticeably.

Figure 9 presents the average precision and recall of RRF, ARF-60D, IRF, ADA and SVM for each iteration, while Figure 10 compares their efficiency. From Figures 9 and 10, we can draw the following conclusions:

- RRF is too computationally intensive for relevance feedback, since its echo time grows linearly with the number of iterations. On the scopes of 80 and 120, the echo time of RRF can go as high as 20 to 30 seconds. In comparison with ARF-60D and IRF, RRF runs 2 to 3 times slower than the latter, but only improves the precision and recall marginally by about $1-3\%$.
- ARF-60D dramatically outperforms both ADA and SVM on retrieval performance. It improves the precision and recall over ADA by $24-34\%$. As to the efficiency, ARF-60D runs as efficiently as ADA on the scope of 120, but runs about 10% faster than the latter on the scopes of 20, 40 and 80.
- ARF and IRF consistently achieves comparable retrieval performance against RRF. Although RRF attains the best precision/recall in all cases, ARF-60D performs as good as RRF on the scope of 20, and is only slightly inferior to the latter on the scopes of 40, 80 and 120. Moreover, ARF-60D consistently outperforms IRF, while IRF improves the retrieval performance noticeably over ADA and SVM in all cases.
- SVM can not achieve good performance with small training samples. On the scopes of 20 and 40, SVM severely deteriorates the precision versus ADA by $7-10\%$. Only with more training examples from the scopes of 80 and 120, SVM achieves comparable retrieval performance against ADA. It degrades the precision and recall versus ADA slightly by 1% on the scope of 80, while improving them over ADA by 1% on the scope of 120.

Scope	ARF-179D			ARF-120D			ARF-60D			IRF		
	P(%)	R(%)	Echo time (sec.)	P(%)	R(%)	Echo time (sec.)	P(%)	R(%)	Echo time (sec.)	P(%)	R(%)	Echo time (sec.)
20	58.79	–	2.90	59.04	–	3.35	58.79	–	3.10	56.71	–	2.38
40	56.35	–	5.50	56.73	–	6.02	56.39	–	5.08	56.56	–	4.15
80	51.16	39.48	8.94	51.17	39.53	9.92	52.72	40.58	7.96	51.66	39.24	9.90
120	42.51	49.67	10.61	42.41	49.52	10.66	45.21	51.73	10.29	43.24	49.76	11.80

Table 3 Performance of ARF-179D, ARF-120D, ARF-60D and IRF on the $10th$ iteration, where P/R denotes precision/recall; dash sign (–) indicates the values that are not used for performance evaluation.

– During online learning, our dynamic feature extraction is very successful in extracting the optimal features. By cooperating with our *adaptive sample selection* approach, it can remove up to 70% of all features with no more than 3% degradation on precision and recall.

Figure 10 demonstrates that the echo time of IRF drops noticeably at some point (where the number of training examples exceeds the threshold N_0). After that, the echo time only increases slightly with the iteration number, since IRF will employ BRSR to perform sample reduction. Figure 10 (a) also illustrates that, on the scope of 20, the echo time of ADA and SVM decreases slightly with the number of iterations. This is because neither ADA nor SVM can yield a strong classifier using small training samples (from the first several iterations). With more training examples available in latter iterations, ADA and SVM are able to train stronger classifiers, which improve the efficiency by excluding more database objects from the nearest neighbor search.

Experimental results presented so far demonstrate that our method achieves remarkable improvement on precision and recall over ADA and SVM. By combining multiple tree classifiers, our method can find multiple nonlinear clusters of relevant objects, while ADA only performs optimally in unimodal-Gaussian case. On the other hand, our method can train a strong classifier from small training samples, so it dramatically outperforms traditional classifiers such as SVM.

Figure 11 presents the precision-recall curves of ARF-60D, IRF, ADA and SVM on difference scopes. This figure demonstrates the impact of sample size on the retrieval performance of compared methods. From this figure, we can draw the following observations:

– All methods that can handle multimodality (such as ARF, IRF and SVM) achieve noticeable gain in performance as the scope increases. To attain the same recall, they can maintain much higher precision on larger scopes. Since more training examples are available on larger scopes, this indicates that these methods perform better with larger training samples.
– ADA, which assumes unimodal-Gaussian distribution for both relevant objects and irrelevant ones, does not fulfill obvious improvement in performance with larger training samples, since its precision-recall curves (of all scopes) almost overlap with each other.

The above observations confirm our previous research results [31] that relevant objects distribute multimodally in the feature space. To payback the user's efforts on providing more training examples, a relevance feedback method must be able to address the multimodal distribution (of multimedia objects).

5 Further Discussions

During multimedia information retrieval, our system assumes a so-called *greedy user model* [36]. That is, our system always shows the *most-relevant* (i.e., top k retrieved) objects to the user. On the other hand, some researchers [29] employ an active learning algorithm and only prompt the most uncertain objects for the user to annotate. The active learning algorithm achieves the most gain in performance when the *most-relevant* objects are not the *most-informative* ones [36]. For example, if multiple iterations are run, some of the *most-relevant* objects may be strongly correlated with previously-labeled objects. However, in general case, this scenario is not the key issue in multimedia retrieval. As indicated by our experiments (cf. Section 4.5), if only 20 objects are returned and checked by the user in each iteration, on average, there will still be quite a few irrelevant objects after multiple iterations are run. Since we train a strong classifier (i.e., online random forests) in each iteration, most newly returned irrelevant objects must be different from those known irrelevant ones. These irrelevant objects are definitely among the *most-informative* ones, since they are mistakenly classified as relevant. So, it is always favorable to prompt the *most-relevant* objects to the user, as long as they have not been annotated. Unfortunately, the irrelevant training sample (accumulated from multiple iterations) can quickly become oversized for training a regular random forest online, so it is necessary to reduce the size of (irrelevant) training sample with our sample selection methods (i.e., biased random sample reduction and/or adaptive sample selection). However, it is still possible to integrate the active learning algorithm with the *greedy user model*. For instance, Tieu *et al.* [28] combined the two together by asking the user to label both *most-relevant* objects and *most-uncertain* ones. In this case, as pointed out by Zhou *et al.* [36]: *the question is how do we strike a balance between the two optimally?* We will study how to integrate the active learning technique into our method in the future.

In the presented method, similar to most existing relevance feedback approaches [24, 28, 29, 35], we focus on challenging the *online learning* problem. That is, during relevance feedback, our system does not resort to any off-line processing or pre-knowledge about the data. It just iteratively

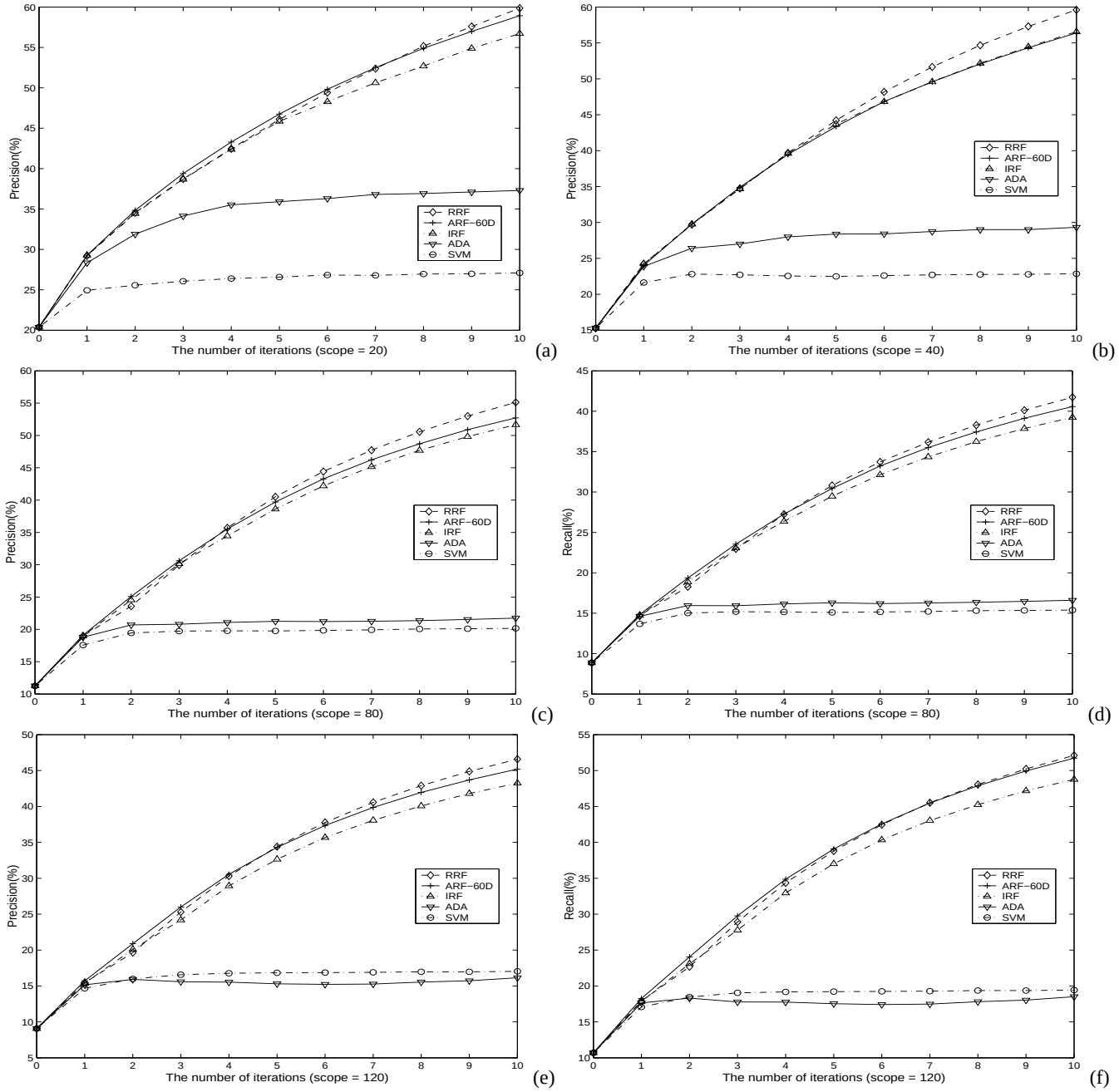


Fig. 9 Performance comparison of RRF, ARF-60D, IRF, ADA and SVM. (a) and (b) illustrate the precisions for scopes of 20, 40. (c) and (d) present the precision and recall for the scope of 80. (e) and (f) demonstrate the precision and recall for the scope of 120.

captures the user's query concept(s) from its interaction with the user. For unseen queries, there are no other choices but to conduct an online learning. Hence, it is always a critical issue to improve the online learning performance of relevance feedback. Nonetheless, we can still boost the efficiency of relevance feedback using appropriate off-line processing and pre-knowledge about the data. For example, Bartolini *et al.* [13] proposed a relevance feedback bypass technique, where they suggest accumulating a mapping from a query point to an optimal query and feature weights. For an incoming query, their system might bypass the relevance feedback process by directly calculating the optimal query and feature weights. Similar techniques can be integrated with our method to reduce the interaction times between the system and the user. In the future, we will perform detailed research on this topic.

Although we only evaluate our method on an image database, our interactive pattern analysis is also applicable to other multimedia data (such as text and speech), as long as they are indexed by real-valued feature vectors. For example, text documents are also indexed by high-dimensional feature vectors, and relevance feedback has been extensively addressed in text-based information retrieval [1]. In addition, with the

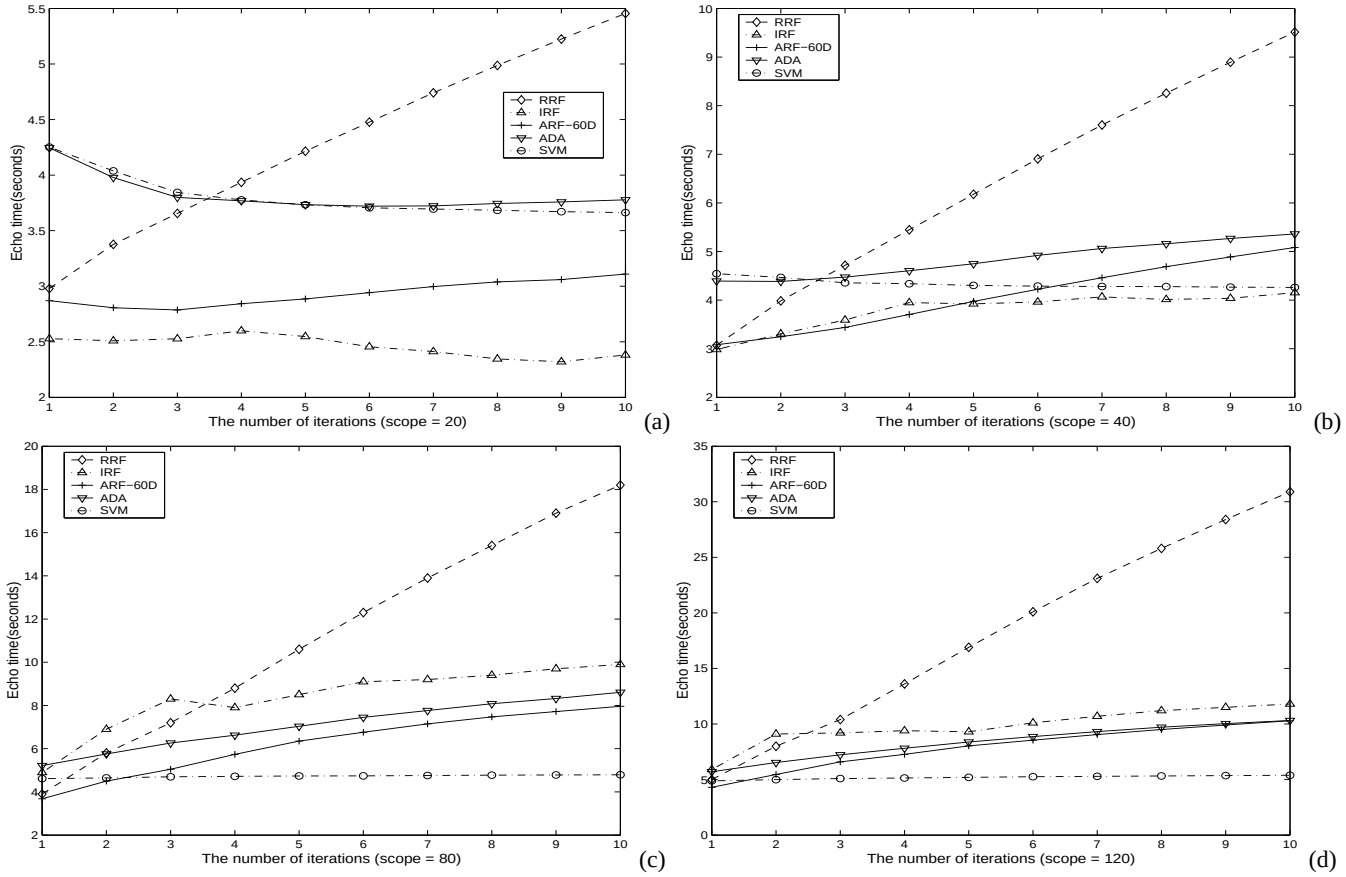


Fig. 10 Effi ciency of different methods. (a)/(b)/(c)/(d) illustrate the echo time for the scopes of 20/40/80/120, respectively.

proven good performance of our *dynamic feature extraction* method, the presented pattern analysis method is an extremely favorable choice for searching high-dimensional multimedia databases (such as text databases, etc.). In case each database object comprised multiple multimedia items (such as text document, speech data and image, etc.), we can concatenate their feature vectors together and directly apply our method on the composite feature space. A more elegant solution is to apply our method to each feature space separately. We can then obtain the final retrieval result by combining the search results from all the feature spaces. For example, the overall classified-relevant set should be the intersection of the classified-relevant sets from all the feature spaces.

6 Conclusions

In this paper, we presented an interactive pattern analysis method for multimedia retrieval. With our method, the user is allowed to participate in the retrieval loop by labeling the retrieved objects as relevant or irrelevant. Using user-labeled objects as training examples, our method employs an online pattern analysis technique to learn the distribution pattern of relevant objects; and it intelligently reduces irrelevant objects from the database. From the reduced object set, our approach refines its previous retrieval results by returning the top-k nearest neighbors (of the query) to the user. To perform interactive pattern analysis, we employ a pattern classification algorithm called random forests, which is a composite classifier trained from multiple tree classifiers.

To generalize the regular random forests for interactive pattern analysis, we present two online pattern classification algorithms termed as interactive random forests and adaptive random forests. By combining random forests with a novel two-level resampling method called biased random sample reduction, our interactive random forests achieves noticeable gain in efficiency over the regular random forests. To further boost the effectiveness of our interactive pattern analysis for relevance feedback, our adaptive random forests combines random forests with two adaptive learning techniques termed as dynamic feature extraction and adaptive sample selection. Dynamic feature extraction aims to alleviate the curse of dimensionality by extracting a subspace (of the original feature space) using balanced information gain, while adaptive sample selection improves the efficiency of pattern analysis by choosing the most-informative examples for relevance feedback. With adaptive sample selection, the adaptive random forests performs the sample selection more effectively by distinguishing between the *most-informative* training examples and the *less-informative* ones. Moreover, by using dynamic feature extraction, it not only further boosts the efficiency of relevance feedback by removing the redundancy in the fea-

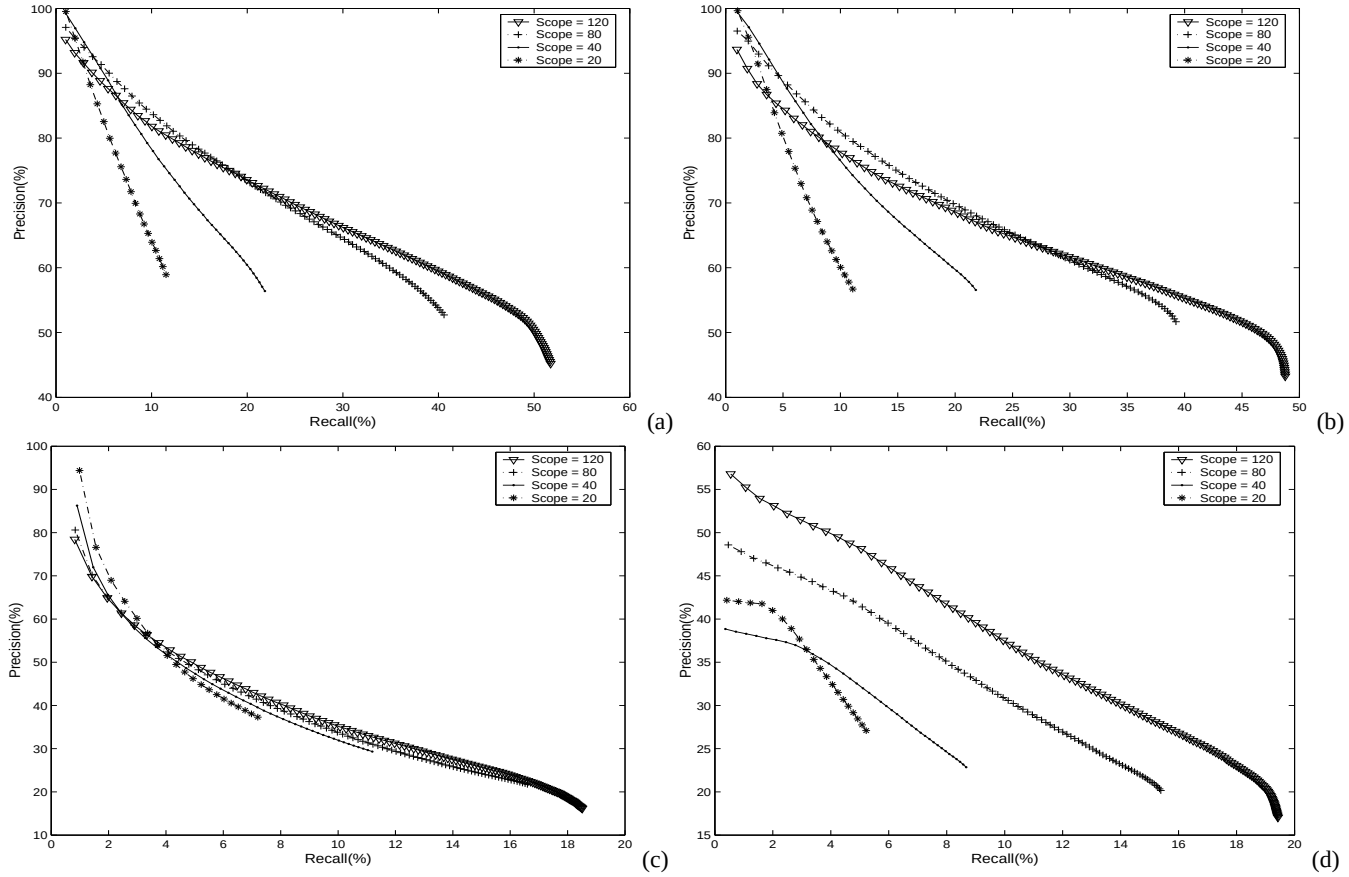


Fig. 11 Precision-recall curves of compared methods. (a)/(b)/(c)/(d) demonstrate the precision-recall curves of ARF-60D/IRF/ADA/SVM, respectively.

ture space, but also accommodates the ability to overcome the curse of dimensionality. Our experiments indicate that ARF is more cost-effective than both IRF and RRF.

References

1. R. Baeza-Yates and B. Ribiero-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.
2. I. Bartolini, P. Ciaccia, and F. Waas. Feedbackbypass: A new approach to interactive similarity query processing. In *Intl. Conf. on VLDB*, Oct.31-Nov.2 2001.
3. S. Brandt, J. Laaksonen, and E. Oja. Statistical shape features in content-based image retrieval. In *Proceeding of ICPR*, Sept. 2000.
4. L. Breiman. Bagging predictors. *Machine Learning*, 24:123 – 140, 1997.
5. L. Breiman. Random forests–random features. Technical Report 567, Department of Statistics, University of California,Berkeley, September 1999.
6. L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.
7. M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *Proc. of ACM Symposium on Theory of Computing*, 1997.
8. I. J. Cox, M. L. Miller, T. Minka, T. V. Papathomas, and P. N. Yianilos. The beyesian image retrieval system, pichunter: Theory, implementation and psychophysical experiments. *IEEE Conf. Computer Vision and Pattern Recognition*, XX(Y), 2000.
9. T. Elomaa and J. Rousu. General and efficient multisplitting of numerical attributes. *Machine Learning*, 36(3):1 – 49, 1999.
10. B. Enfron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.
11. Y. Freund and R. Shapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and Systerm Science*, 55(1):119-139, 1997.
12. K. Fukunaga. *Introduction to Statistical Pattern Recognition*. San Deigo, California: Academic Press, Inc., 1990.
13. G. D. Guo, A. K. Jain, W. Y. Ma, and H. J. Zhang. Learning similarity measure for natural image retrieval with relevance feedback. *IEEE Conference on CVPR*, 1:731 – 735, June 2001.
14. Y. Ishikawa, R. Subramanya, and C. Faloutsos. Mindreader: Query databases through multiple examples. In

Proc. of the 24th VLDB Conference, 1998.
15. T. Joachims. *Advances in Kernel Methods - Support Vector Learning, Chapter 11*, chapter Making Large-Scale SVM Learning Practical, pages 41–56. MIT Press, 1998.
16. I. Kononenko. On biases in estimating multi-valued attributes. In *Proc. of the Fourteenth International Joint Conf. on Artificial Intelligence*, 1995.
17. W. Y. Ma and H. Zhang. *Handbook of Multimedia Computing*, chapter Content-Based Image Indexing and Retrieval, pages 19–20. London: Routledge, 1998.
18. S. D. MacArthur, C. Brodley, and C. Shyu. Relevance feedback decision trees in content-based image retrieval. In *Proc. IEEE Workshop on CAIVL*, 2000.
19. T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.
20. W. Niblack et al. The qbic project: Querying images by content using color, texture, and shape. In *Storage and Retrieval for Image and Video Databases*. SPIE, 1993.
21. G. Pass, R. Zabih, and J. Miller. Comparing images using color coherence vectors. In *Proceedings of ACM Multimedia 96*, pages 65–73, Boston MA USA, 1996.
22. K. Porkaew, K. Chakrabarti, and S. Mehrotra. Query refinement for multimedia similarity retrieval in mars. In *Proc. of ACM Multimedia*, Novemember 1999.
23. J. Quinlan. Introduction of decision tree. *Machine Learning*, 1:81 – 106, 1986.
24. Y. Rui and T. Huang. Optimizing learning in image retrieval. *IEEE Conf. on CVPR*, June 2000.
25. Y. Rui, T. Huang, and S. Mehrotra. Content-based image retrieval with relevance feedback in mars. In *Proc. IEEE Int. Conf. on Image Proc.*, 1997.
26. J. R. Smith and S. F. Chang. Transform features for classification and discrimination in large image databases. In *Proc. IEEE ICIP.*, 1994.
27. Z. Su, S. Li, and H. Zhang. Extraction of feature subspaces for content-based image retrieval using relevance feedback. In *Proc. of ACM Multimedia*, 2001.
28. K. Tieu and P. Viola. Boosting image retrieval. In *IEEE Int'l Conf. Computer Vision and Pattern Recognition (CVPR'00)*, June 2000.
29. S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *Proc. of ACM Multimedia*, 2001.
30. N. Vasconcelos and A. Lippman. Bayesian relevance feedback for content-based image retrieval. In *Proc. IEEE Workshop on CAIVL*, 2000.
31. Y. Wu and A. Zhang. Category-based search using metadatabase in image retrieval. In *Proc. IEEE Int'l Conf. on Multimedia and Expo (ICME'02)*, 2002.
32. Y. Wu and A. Zhang. A feature re-weighting approach for relevance feedback in image retrieval. In *Proc. IEEE Int'l Conf. on Image Processing (ICIP'02)*, 2002.
33. Y. Wu and A. Zhang. Interactive pattern analysis for searching multimedia databases. In *Proc. 8th Int'l Workshop on Multimedia Information Systems (MIS'02)*, Oct.31-Nov.2 2002.
34. Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proc. of ICML-97*, 1997.
35. X. S. Zhou and T. S. Huang. Comparing discriminating transformations and svm for learning during multimedia retrieval. In *Proc. of ACM Multimedia*, Sept. 2001.
36. X. S. Zhou and T. S. Huang. Relevance feedback for image retrieval: A comprehensive review. *ACM Multimedia Systems*, 8(6):536 – 544, 2003.