Multimedia Systems 9: 131–143 (2003)
Digital Object Identifier (DOI) 10.1007/s00530-003-0084-5

Multimedia Systems

Adaptive tree similarity learning for image retrieval*

Tao Wang[1], **Yong Rui**[2], **Shi-min Hu**[1], **Jia-guang Sun**[1]

[1] Department of Computer Science and Technology, Tsinghua University Beijing 100084, P. R. China
e-mail: wangtao@cg.cs.tsinghua.edu.cn, {shimin, sunjg}@mail.tsinghua.edu.cn

[2] Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
e-mail: yongrui@microsoft.com

Abstract. Learning-enhanced relevance feedback is one of the most promising and active research directions in content-based image retrieval in recent years. However, the existing approaches either require prior knowledge of the data or converge slowly and are thus not coneffective. Motivated by the successful history of optimal adaptive filters, we present a new approach to interactive image retrieval based on an adaptive tree similarity model to solve these difficulties. The proposed tree model is a hierarchical nonlinear Boolean representation of a user query concept. Each path of the tree is a clustering pattern of the feedback samples, which is small enough and local in the feature space that it can be approximated by a linear model nicely. Because of the linearity, the parameters of the similartiy model are better learned by the optimal adaptive filter, which does not require any prior knowledge of the data and supports incremental learning with a fast convergence rate. The proposed approach is simple to implement and achieves better performance than most approaches. To illustrate the performance of the proposed approach, extensive experiments have been carried out on a large heterogeneous image collection with 17,000 images, which render promising results on a wide variety of queries.

Key words: Content-based image retrieval (CBIR) – Adaptive filter – Linear similartiy model – Tree similarity model

1 Introduction

In the past decade, both technology push (e.g., multimedia data analysis and machine learning [5,20,24,27,29,34,36]) and application pull (e.g., various digital library [3,8]) have contributed to the proliferation of image retrieval techniques [7,25,29]. However, even after years of extensive research, helping users find their desired images accurately and quickly is still an open problem.

In the early years of content-based image retrieval (CBIR), most researchers devoted their efforts to finding the best visual features or the best similarity measure [11,12,15,17,18,32]. However, because of the complexity and subjectivity of the visual content, no single feature can discriminate all images, nor can a single similarity measure meet every user's need. On the other hand, according to recent user study results [23], what average users really want are the systems that support queries based on high-level concepts (e.g., red apples on a brown table), not low-level features (e.g., 30% red color with 50% brown color). Mapping from the low-level visual features to high-level concepts dynamically has emerged as the focus of CBIR [25].

Because each user may interpret an image differently, it is necessary and effective to have the user in the retrieval loop in order to discover the user's query concept, which is important for dynamically constructing better mappings between low-level visual features and high-level concepts [39]. For example, given a query image with a person standing in a sunset scene, it would be difficult for the CBIR system to guess whether the user's query is to search the person or the sunset scene. Only after the user gives a few relevant and/or irrelevant images will the system know what the user really wishes to query. Motivated by this observation, several teams have introduced relevance feedback into CBIR systems to adapt image features and similarity measures to best reflect high-level concepts based on samples provided by users. Relevance feedback was first developed in the text-based information retrieval (IR) research community [27], which has been gaining considerable momentum for CBIR in recent years [25].

Relevance feedback techniques are based on different learning mechanisms and similarity models. Among various proposed feedback approaches, suggested ones include the probabilistic Bayesian approach [2,4,22,34], the transductive learning approach (e.g., discriminate EM) [38], the boosting approach [31], the kernel approximation approach (e.g., support vector machines) [30,32,39], and the optimization learning (OPL) approach [24]. While the various approaches have improved the performance of CBIR, many of them suffer from one or more of the following limitations:

* An early version of part of the system was reported in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition* 2001.

- The learning process has its best strength only if prior knowledge about the data distribution or a large training set is available [4,22,30,34,38].
- If users do not give sufficient feedback samples during the retrieval process, only a suboptimal solution of the similarity model can be achieved for the parameter estimation by the *minimum mean square error* (MMSE) criterion [24].
- The convergence to the optimal solution of the similarity model is slow and the computation cost is high [9,22,38].
- While the linear similarity model proves easy to implement and fast in computation [26], it is not sufficient to model the nonlinearity of human vision perception.

All the above limitations prevent the existing approaches from being fully deployed in practical systems. In this paper, we propose a novel relevance feedback technique based on adaptive filtering and a Boolean tree similarity model. Adaptive filters have been successfully used for more than four decades in various research areas including signal processing, automatic control, and system identification [27]. The adaptive filters are rooted in the optimal estimation theory, whose input X(n) and output error $e(n)$ drive it to automatically adapt to the MMSE solution of the estimated model's parameters efficiently without any prior knowledge about data distribution. Considering the human vision system as an intelligent signal filter (a black box with input of image feature and output of similarity), it is possible to use the optimal adaptive filter to better adaptively estimate the parameters of the visual similarity model. By doing so, we can apply the least mean square (LMS) and recursive least square (RLS) adaptive filter algorithms to solve the learning problem in a theoretical way. They result in simple implementation, fast convergence rate, and good performance.

The conventional adaptive filter assumes that the unknown visual perception system is a linear similarity model. The similarity model for human vision perception, however, is a complex nonlinear system [11]. Furthermore, we extend the conventional adaptive filter to a nonlinear tree similarity model. The proposed tree similarity model is a hierarchical representation of clustering patterns of all feedback samples. It decomposes a user's complex query concept into a Boolean combination of multiple simpler subconcepts. Each pattern of many possible subconcepts defines a path leading from the tree root node to a leaf node. All the subconcepts are then combined using Boolean algebra to yield the overall query concept of the user. The benefits of using this proposed tree model are:

- Compared with the overall query concept, each path of the tree spans smaller feature subspace and therefore can be well approximated as a linear model on which the optimal adaptive filter can be better implemented.
- Because the overall query is modeled as a nonlinear Boolean combination of all the subconcepts, it does not sacrifices the nonlinearity.
- Some image features are suitable for quickly filtering out irrelevant images at a coarse level, while others are suitable for fine-tuning the image similarity. The tree model renders itself to such a coarse-to-fine hierarchical search such that it improves the retrieval speed.

The remainder of the paper is organized as follows. In Sect. 2, we first introduce various important concepts and notations of CBIR before going into the details of the proposed approach. In Sect. 3, we focus on the optimization learning approach (OPL) [24] for related work, which is one of the available advanced approaches. In Sect. 4, we give a detailed description of the proposed adaptive learning approaches based on optimal adaptive filters. We further discuss how to solve the learning order issue encountered in CBIR and give computation complexity comparisons between OPL, LMS, and RLS. In Sect. 5, we extend adaptive-filtering-based relevance feedback techniques from a linear model to a nonlinear adaptive tree similarity model. Specifically, we show how to train the CBIR system to adaptively learn the tree similarity model online, calculate the overall similarity from the tree, and efficiently retrieve images based on this model. In Sect. 6, extensive experiments over a large heterogeneous image collection with 17,000 images are reported to evaluate the retrieval performance of the adaptive tree similarity model. Concluding remarks are given in Sect. 7.

2 Concepts and notations

Before we consider the details of the paper, it would be beneficial to first introduce and define some important concepts and notations that will be used in the paper.

Let I be the number of features we are studying and M the total number of images in the database. We use $\vec{F}_{mi} = [f_{mi1}, f_{mi2}, \ldots, f_{miK_i}]^T$ to denote the i-th feature vector of the m-th image, $m = 1, 2, \ldots, M$, where K_i is the dimension of the i-th feature vector. For example, for a six-element color moment feature vector, $K_i = 6$.

Let $\vec{F}_{qi} = [f_{qi1}, f_{qi2}, \ldots, f_{qiK_i}]^T$ denote the i-th feature vector of the query image q. We further define a difference vector between image $m, m = 1, 2, \ldots, M$, and the query image q as:

$$\vec{X}_i(m) = [|f_{mi1} - f_{qi1}|, \quad \ldots, \quad |f_{miK_i} - f_{qiK_i}|\,]^T \qquad (1)$$

where $|x - y|$ represents the difference between x and y. Because different feature elements may have different contributions to the perception of image content, different weights can be associated with the feature elements to reflect this effect [24]. The distance between image $m, m = 1, 2, \ldots, M$ and query q, in terms of the i-th feature, can therefore be calculated as:

$$\begin{aligned} g_i(m) &= \vec{X}_i(m)^T W_i \vec{X}_i(m), \quad L_2 \text{ metric norm} \\ g_i(m) &= \vec{W}_i^T \vec{X}_i(m), \qquad\quad L_1 \text{ metric norm} \end{aligned} \qquad (2)$$

depending on if we want to use L_1 or L_2 distances. For L_2 distance, W_i is a weight matrix, while for L_1 distance, $\vec{W}_i$ is a weight vector.

After we have discussed how to compute image distances based on an individual feature, the overall distance $d(m)$ based on multiple features can be computed in two ways. One way is to not differentiate the difference among features and stack all the feature elements (from all the individual features) into a big overall feature vector to compute $d(m)$. This approach was used in most of the existing systems. Because this model has no hierarchy, we refer to it as the "flat model" in this paper. Another way is to construct a hierarchical model, where the

overall distance $d(m)$ is defined as:

$$d(m) = U(g_i(m)) \tag{3}$$

where $U(.)$ is a linear function that combines the individual distances $g_i(m)$ to form the overall distance $d(m)$. We will refer to this model as the "hierarchical model."

While we have discussed how to compute the *distance* between two images, *similarity* is more commonly used in CBIR. To convert between distance and similarity, we adopt the approach proposed in [9]. Assuming the distance $d(m)$, $m = 1, 2, \ldots, M$ obeys the Gaussian distribution of $N(0, \sigma^2)$, the similarity $\pi(m)$ between image m and query image q is the likelihood of this distribution, with $\pi(m) = 0$ being the least similar and $\pi(m) = 1$ being the most similar:

$$\begin{aligned} \pi(m) &= \exp(-\frac{d(m)^2}{2\sigma^2}), \quad \pi(m) \in [0, 1] \\ d(m) &= \sqrt{-2\sigma^2 \ln(\pi(m))} \end{aligned} \tag{4}$$

Similarly, we have the following relationships for the i-th feature:

$$\begin{aligned} \lambda_i(m) &= \exp(-\frac{g_i(m)^2}{2\sigma^2}), \quad \lambda_i(m) \in [0, 1] \\ g_i(m) &= \sqrt{-2\sigma^2 \ln(\lambda_i(m))} \end{aligned} \tag{5}$$

3 Related work

Among the existing approaches [4,10,13,14,20,22,24,26,30,31,38] introduced in Sect. 1, we mainly compare the optimal learning (OPL) approach [23] with our approach, for the following reasons:

- OPL is one of the best techniques available; it formulates the relevance feedback in a vigorous optimization framework and derives *explicit* optimal solutions for the weights in a linear model. Thus it performs better than many other existing MARS [26] and Mind Reader [10] approaches.
- Unlike many other approaches that are tested only on preselected queries over small data sets, the OPL approach has been tested with a wide variety of 400 random queries over a large 17,000 heterogeneous image collection. In this manner, it obtains an objective evaluation of the various retrieval techniques.

The OPL approach can be summarized as follows. Let N be the number of feedback images (training samples), and let $\pi(n)$ be the degree of relevance for training sample $n, n = 1, \ldots, N$ given by the user. The overall distance between a training sample and a query is defined as:

$$\begin{aligned} d(n) &= \sum_{i=1}^{I} u_i g_i(n) \\ g_i(n) &= \vec{X}_i^T(n) W_i \vec{X}_i(n) \end{aligned} \tag{6}$$

where W_i is the low-level (interfeature) weights and u_i is the high-level (intrafeature) weights. The OPL approach derives the optimal query vector $\vec{F}_{qi}$ and optimal weights by using the Lagrange multipliers [24]:

$$\begin{aligned} \vec{F}_{qi} &= \frac{\sum_{n=1}^{N} \pi(n) \vec{F}_{ni}}{\sum_{n=1}^{N} \pi(n)} \\ W_i &= \begin{cases} (\det(C_i))^{1/K_i} C_i^{-1}, & \det(C_i) \neq 0 \\ \mathrm{diag}(1/\sigma_1^2, 1/\sigma_2^2, \ldots 1/\sigma_{K_i}^2), & \text{otherwise} \end{cases} \\ u_i &= \sum_{j=1}^{I} \sqrt{\sum_{n=1}^{N} \pi(n) g_j(n) / \sum_{n=1}^{N} \pi(n) g_i(n)} \end{aligned} \tag{7}$$

The term C_i is the weighted K_i-by-K_i covariance matrix of the i-th feature vector of the feedback samples.

$$C_i = \frac{\sum_{n=1}^{N} \pi(n) (\vec{X}_i(n) \vec{X}_i^T(n))}{\sum_{n=1}^{N} \pi(n)}$$

When computing W_i, the OPL approach switches between a full matrix and a diagonal matrix depending on the relationship between the number of feedback samples N and the length of the feature vector K_i. When N>K_i, the OPL uses the full matrix form to take advantage of large feedback samples. When $N < K_i$, the OPL uses the diagonal matrix to avoid noisy parameter estimation. The OPL approach has many advantages over other existing approaches, including MARS [26] and Mindreader [10]. However, it still has the following difficulties:

- Calculating det(C_i) and C_i^{-1} is quite expensive, i.e., requires $O(\sum_i^I (K_i)^3)$ operations, which is not desirable for practical image retrieval applications;
- When the number of feedback samples N is small, e.g., $N < K_i$, the weights W_i reduce from a full matrix to a diagonal matrix, which results in suboptimal solutions for the parameter estimation of the similarity model.
- OPL is a batch learning approach, which requires that all samples be given simultaneously before it can learn. When an additional feedback example is presented, there is no easy way to incrementally incorporate the new example without recomputing the weights.
- More importantly, OPL uses a linear similarity model to combine multiple feature similarities with the overall similarity, which is too restrictive and not sufficient in modeling human vision perception.

The next two sections try to resolve the above difficulties. In Sect. 4, we propose an adaptive-filter-based learning approach. It converges quickly to current optimal feature weights for a linear similarity model and avoids expensive computation by using a recursive incremental learning paradigm. In Sect. 5, we propose an adaptive tree similarity model to deal with the linear limitation in OPL. The adaptive filter performs better in the adaptive tree model because the divided feature subspace is approximated better as a linear model in each path of the tree.

4 Learning with adaptive filters

The human vision system can be considered as a black box (see Fig. 1). For query image q and feedback image n,

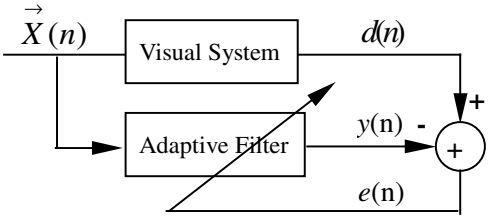

Fig. 1. Adaptive-filter-based feedback model

$n = 1, 2, \ldots, N$, the input to the filter is the difference feature vector $\vec{X}(n)$, and the output from the filter is the similarity distance $d(n)$. Although we do not know the human vision system's response function to $\vec{X}(n)$, it is possible for us to construct an adaptive filter to estimate it according to user feedback. Let the input to the adaptive filter be the same as the input to the human vision system, i.e., $\vec{X}(n)$ and the output of the adaptive filter be $y(n)$. By comparing $y(n)$ against $d(n)$, we can obtain an error signal $e(n)$, which can then be used to drive the adaptive filter to adjust model parameters in order to better simulate the human vision system's response function (see Fig. 1). In this section, we first give the optimal Wiener solution and then develop two relevance feedback techniques based on LMS and RLS. We further discuss how to solve the learning order issues encountered in CBIR and give computation complexity comparisons between OPL, LMS, and RLS.

4.1 Optimal Wiener filter

Assuming a linear system $d(n) = \vec{W}^T \vec{X}(n)$ and given a wide-sense stationary (WSS) input signal $\vec{X}(n)$, output signal $y(n)$, and desired output signal $d(n)$, the Wiener filter is the optimal stochastic filter that minimizes the variance of the error [27]:

$$\begin{aligned}\min_W E[e^2] &= E[(d(n) - \vec{W}^T(n)\vec{X}(n))^2] \\ &= E[d(n)^2] - 2\vec{P}^T\vec{W} + \vec{W}^T R_{xx}\vec{W} \end{aligned} \tag{8}$$

where $\vec{W} = [w(0), w(1), \ldots, w(K-1)]^T$ is the filter coefficient, K is the length of the Wiener filter and

$$R_{xx} = E[\vec{X}(n)\vec{X}^T(n)], \quad \vec{P} = E[d(n)\vec{X}(n)] \tag{9}$$

The gradient of $E[e^2]$ with respect to the filter coefficient is

$$\nabla = \partial(E[e^2])/\partial\vec{W} = -2\vec{P} + 2R_{xx}\vec{W} \tag{10}$$

By setting the gradient to zero, we arrive at the optimal Wiener solution:

$$\vec{W} = R_{xx}^{-1}\vec{P} \tag{11}$$

This solution is great in theory, but in reality we do not know the statistics R_{xx} and P of the signals a priori. Fortunately, we can estimate the statistics on the fly while we are computing the optimal solution. Two techniques are LMS and RLS, with LMS approximating the steepest gradient descent and RLS approximating R_{xx} and P directly.

4.2 Least mean square solution (LMS)

Because we do not know R_{xx} and P in advance, the gradient descent approach can be used to solve the nonlinear optimization problem $\min_W E[e^2]$. At each iteration we compute the gradient and move the solution toward the steepest gradient descent direction, i.e.:

$$\nabla^{(n)} = \partial E(e^2)/\partial\vec{W}^{(n)}] = -2e(n)\vec{X}(n) \tag{12}$$

$$\vec{W}^{(n+1)} = \vec{W}^{(n)} - \mu\nabla^{(n)} \tag{13}$$

where μ is the step size and n is the iteration index. The LMS algorithm was developed four decades ago by Widrow and Hoff [36]. Today it is still widely used because of its simplicity, low computation demands, and great performance [27]. Next we give a complete LMS algorithm developed for relevance feedback in CBIR.

[Procedure 1: LMS relevance feedback algorithm]

(A) Initialization:
Choose step size $0 < \mu < 2$ and set filter coefficients to

$$\vec{W}(0) = [1/K, 1/K, \ldots, 1/K] \tag{14}$$

(B) For each $n = 1, 2, \ldots, N$:

(a) Compute the distance $y(n)$ based on the current weights

$$y(n) = \vec{W}^T(n)\vec{X}(n) \tag{15}$$

(b) Compute the error signal

$$e(n) = d(n) - y(n) \tag{16}$$

Note that compared with standard Wiener filters, we have an extra step to convert from the similarity to distance $d(n)$.

(c) Compute the updated weights

$$\begin{aligned}&\vec{W}(n) \\ &\quad = \vec{W}(n-1) + \frac{\mu}{a + \vec{X}^T(n)\vec{X}(n)}\vec{X}(n)e(n)\end{aligned} \tag{17}$$

where a is a small positive constant to avoid denominator being 0.

The LMS-based relevance feedback algorithm is elegant in theory, easy to Implement, and requires very little computation.

4.3 Recursive least square algorithm (RLS)

Instead of approximating the gradient, RLS attempts to approximate the R_{xx} and P directly. It uses the famous matrix inverse equation [27]

$$\begin{aligned}A &= B^{-1} + CF^{-1}C^T \Rightarrow A^{-1} \\ &= B - BC(F + C^T BC)^{-1}C^T B\end{aligned}$$

to simplify the computation of R_{xx}^{-1}. For a detailed derivation of the RLS algorithm, please refer to Appendix A. Compared with LMS, RLS has the following features:

- Because LMS uses the noisy gradient to approximate the true gradient, it converges fast at initial steps and gradually slows down and even oscillates when approaching the final solution. RLS, on the other hand, estimates R_{xx} and P directly at each iteration, thus resulting in faster overall convergence.
- However, RLS's faster convergence speed is at the cost of more computation. In addition, when training samples are not more sufficient than the dimension of the feature vector, estimating R_{xx} and P can be problematic. This may actually result in a slower convergence than LMS in practice.

A detailed comparison between LMS and RLS is given in Sect. 6. We next give the complete RLS algorithm developed for relevance feedback in CBIR.

[Procedure 2: RLS relevance feedback algorithm]

(A) Initialization:

$$\vec{W}(0) = [1/K,\ 1/K, \ldots,\ 1/K], \quad Q(0) = \delta^{-1} I \tag{18}$$

where Q is the inverse of the signal covariance matrix R_{xx} and δ is a small positive constant.

(B) For each $n = 1, 2, \ldots, N$:

1. Compute the distance $y(n)$ based on the current weights

$$y(n) = \vec{W}^T(n)\vec{X}(n) \tag{19}$$

2. Compute the error signal:

$$e(n) = d(n) - y(n) \tag{20}$$

3. Compute the gain vector:

$$\vec{K}(n) = \frac{Q(n-1)\vec{X}(n)}{1/\pi(n) + \vec{X}^T(n)Q(n-1)\vec{X}(n)} \tag{21}$$

4. Compute the updated weights:

$$\vec{W}(n) = \vec{W}(n-1) + \vec{K}(n)e(n) \tag{22}$$

5. Compute the inverse correlation matrix:

$$Q(n) = Q(n-1) - \vec{K}(n)\vec{X}(n)Q(n-1) \tag{23}$$

4.4 Learning orders for adaptive filters in CBIR

In the original adaptive filters, signals $\vec{X}(n)$ and $d(n)$ arrive in a sequential order, while in CBIR there is no explicit order for feedback samples. The two most obvious approaches we can take for this ordering issue are the *forward ordering* approach and the *backward ordering* approach. Let set S contain the N feedback samples in order of decreasing similarity to the query image. That is, the first image in the set has the greatest similarity to the query image and the last image in the set has the least similarity to the query image. The forward approach is to learn the feedback samples from the first to the last, and the backward approach is to learn the samples from the last to the first. Because both LMS and RLS are incremental learning algorithms, we expect the backward approach to be more advantageous: its learning samples are organized in a coarse-to-fine fashion. Just like the hierarchical pyramid approach in optical flow computation [1], the backward approach simulates a hierarchical algorithm to avoid local minimum and to speed up convergence. It saves the "best" example at the last to fine-tune the parameters. We will provide a detailed comparison of the forward and backward learning orders in Sect. 6.1.

4.5 Computation complexity

Given the high computation cost involved in most of today's relevance feedback techniques [10,38], one of our motivations in developing the adaptive-filter-based approach is its efficiency. The OPL approach is already one of the most efficient relevance feedback approaches available, but it still requires $O(K^3 + 2NK^2)$ computations to learn the weights of each item in a K dimension feature vector [23]. If we examine the LMS and RLS algorithms, they only need, respectively, $O(NK)$ and $O(NK^2)$ computations to learn the weights of each item in a K dimension feature vector [27]. As we discussed in Sect. 4.2, LMS is extremely efficient, which means it is linear in both the number of feedback samples N and the feature vector dimension K. Furthermore, unlike OPL, which is a *batch* learning algorithm, both LMS and RLS are *incremental recursive* learning algorithms. That is, when the n-th feedback example becomes available, they can learn it incrementally from example $n-1$, without reexecuting the whole algorithm. To illustrate the difference in their computation complexity, let us plug in some real-world numbers with $K = 18$, $N = 20$. The LMS, RLS, and OPL require 360, 6480, and 18,792 computations, respectively. Both LMS and RLS are more efficient than OPL. It is worth mentioning that LMS is exceptionally efficient; its computation count is an order of magnitude less than RLS's.

5 Tree similarity model

The previous section discussed how the adaptive filters learn the low-level weights W_i in a linear model that is not powerful enough to model the nonlinearity of human visual cognition. Many experiments, e.g., Jain et. al., have shown that the similarity model in human vision is a complex nonlinear model [11]. Furthermore, Krumhansl has proposed a distance density model of similarity in which similarity is assumed to be a function of both the feature difference and the feature density of the feature space [12]. However, because we know little about the density distribution of the feature space and the property of the similarity model due to a limited number of training samples, accurate similarity measures are usually difficult to achieve in reality. Fortunately, we can adopt a "divide-and-conquer" approach whereby a complex nonlinear feature space can be divided into simpler, meaningful, and linear similarity models around the interested clusters in the feature subspace. For example, in Fig. 2, there are three clusters A_1, A_2, A_3 of class A and two clusters, B_1, B_2, of class B, and $\bar{A}$ is the averaged point of all points of class A. If we use the averaged point $\bar{A}$ to

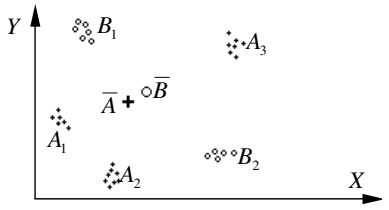

Fig. 2. Many clustering sets for two classes

retrieve all points of class A in a linear model, we will misclassify some points of B to class A when $\bar{A}$ is near class B. So a linear model cannot approximate well the complex nonlinear visual similarity model. Our solution is to divide the nonlinear feature space into a few approximated linear subspaces A_1, A_2, A_3 for class A and B_1, B_2 for class B by clustering these points in the feature space. The adaptive filter therefore can learn better because it assumes the similarity model is linear only in the subspaces A_1, A_2, A_3.

The hierarchical model is widely used to improve the efficiency and precision in pattern recognition [1,16] because it naturally simulates the coarse-to-fine processing. By using a suitable threshold, irrelevant images can be quickly filtered out at top levels and only possible relevant images will be examined at all levels. Another advantage of the hierarchical method is that it avoids the curse of dimensionality (too many features will make the performance worse [33]). If we test more important features earlier in the hierarchy and less important features at a later stage and give different weights to these features, we can avoid the curse of dimensionality to some extent.

The relevance feedback decision tree is broadly used in pattern recognition due to its efficient hierarchical property. It is used for CBIR in [19], which classifies all images into relevant and irrelevant images. Then the similarity is calculated by executing an unweighted K nearest neighbor retrieval on the list. Motivated by the above observations, in our paper, we propose a novel tree similarity model for CBIR (see Fig. 3). Unlike the method in [19], the proposed adaptive tree similarity model is a hierarchical Boolean representation of clustering patterns of all feedback samples (relevant and irrelevant). Because the feature subspace near each path is so small that it can be approximated as a linear model, the adaptive filter is integrated to adapt the feature weights in the tree path. In this section, we will first give a detailed description of how to build and train a tree similarity model. Then we develop a similarity measure algorithm for the adaptive tree similarity model and discuss how to optimize the model's performance.

5.1 Building the tree similarity model

The proposed tree similarity model is a hierarchical representation of clustering patterns of all feedback samples. The sequence of levels from top to bottom is the order of test features. Except for the root level and leaf level of the tree, every level corresponds to one feature test. Therefore, there are $I+2$ levels in the tree model if a CBIR system uses I features. Every node at the same level i is a possible clustering pattern of the corresponding tested i-th feature. Any path from the root to a leaf node is one of the patterns of a user's query concept. Thus all discovered clustering patterns of the feedback samples are represented by paths of the tree after the model has learned a user's feedback. For example, in Fig. 3, C_{ir} denotes the r-th clustering of the i-th feature. At the leaf node, A_j is a clustering label to denote the j-th pattern of class A. Similarly, leaf nodes A_1, A_2, A_3, A_4, A_5 are the discovered patterns of class A, and B_1, B_2 are patterns of class B. The path *root*$\rightarrow C_{11} \rightarrow C_{21} \rightarrow C_{32} \rightarrow A_2$ is a pattern A_2 of relevant images of class A.

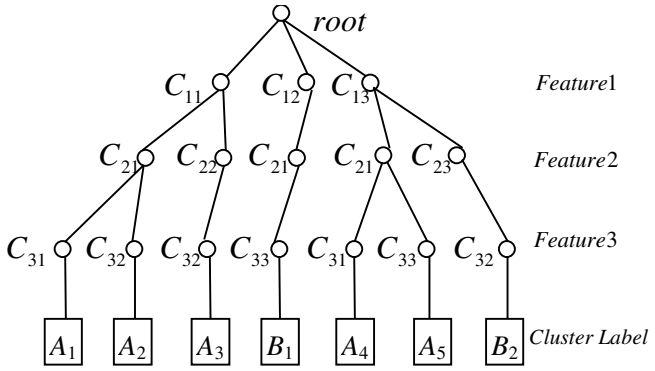

Fig. 3. An adaptive tree similarity model

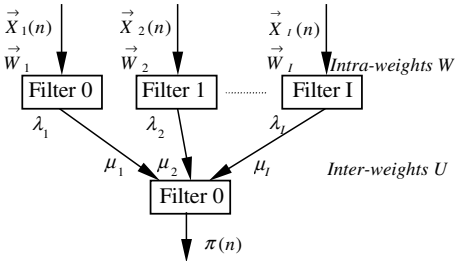

Fig. 4. Learning intrafeature weights W and interfeature weights U by hierarchical filters in a two-level weighting scheme with I features

The tree forms a Boolean model. The relationship between paths is "OR," denoted as $\vee$, for positive samples, or "NOT," denoted as $\neg$, for negative samples. The relationship between all the nodes in a path is "AND," denoted as $\wedge$. The subspace near paths is so small and local that it can be successfully approximated to a linear similarity model. When a user gives feedback samples, the adaptive filtering discussed in Sect. 4 is used to adaptively learn the weights of the linear model in its corresponding path.

The structure of the Boolean model is illustrated in Fig. 4. First, it calculates the similarity $\lambda_i(n)$ of each individual feature by Eq. 5, $i = 1, 2, \ldots, I$. It then uses the Boolean combination to obtain the overall similarity $\pi^*(n)$ between the query image and the n-th image.

$$\pi^*(n) = \bigvee_j(\pi_j(n)) = \begin{cases} \max_j(\pi_j(n)), & j \in S_R \\ 0, & j \in S_N \end{cases}$$

$$\pi_j(n) = \bigwedge_i(g_i(n)) = \sum_{i=1}^{I} u_i \lambda_i(n) \tag{24}$$

where $W = [\vec{W}_1, \vec{W}_2, \ldots, \vec{W}_I]$ is the intrafeature (low-level) weights and $U = [u_1, u_2, \ldots, u_I]$ is the interfeature (high-level) weights, S_R is the pattern set of the relevant images and

S_N is the pattern set of irrelevant images given by a user. Here we use max() to model the "OR" operation and weighted summation to model the "AND" operation. If an image is similar to any pattern in the irrelevant set S_N, its similarity is zero. We use j to indicate a particular path in the tree.

The tree model uses the hierarchical (tree) adaptive filter to learn the interfeature weights and intrafeature weights simultaneously, which is proven to have fast convergence and low computation [37]. Furthermore, the tree model is an *incremental* learning algorithm, which can be updated recursively by adding new paths by or adjusting the feature weights and the center of each cluster when a new sample arrives. We then give a complete algorithm to build and train the tree model from user feedback. The algorithm can learn both intrafeature weights W and interfeature weights U adaptively and update the structure of the tree model recursively after a new sample arrives (see procedures 3 and 4).

5.2 Similarity computation by tree model

After training the tree similarity model by user feedback samples, the total similarity can be calculated using Eq. 24. The whole process is aimed at finding a path with maximum similarity to the query image. If there is no matched path or there is a path belonging to a negative sample, the total similarity is zero. If an image is filtered out by the feature test at a particular level in the path, it exits and keeps the current value as the total similarity. Otherwise, the total similarity is the weighted average of all similarities in each node of the path (see procedure 5). For example, if a user inputs only red cars as relevant feedbacks, the adaptive tree model will cluster these red-car images as query samples and think the user's query concept is *the car with red color*. On the contrary, if a user gives red cars, blue cars, yellow cars, etc. as relevant feedbacks, the tree model can adaptively learn that the user's query concept is the *car* and that the color feature is not important and retrieve many cars with different colors that are similar to any of these clustering feedback images.

5.3 Optimization of the tree similarity model

The proposed tree model is hierarchical. Selecting the order of the tested features is therefore important. The order from coarse to fine is robust and efficient for human vision. A good way to determine which feature is "coarse" and which is "fine" is given according to the information entropy.

Let $p(i|E)$ be the probability that a cluster drawn at random from a set E belongs to class i. Then the entropy or uncertainty of the set E is:

$$H(E) = -\sum_i p(i|E) \log P(i|E) \tag{25}$$

If we perform a feature test T and have k possible outcomes on the patterns in E, we will create k subsets, $E_1, E_2, \ldots, E_k$. The uncertainty about outcome E_j would be:

$$H(E_j) = -\sum_i P(i|E_j) \log P(i|E_j) \tag{26}$$

[Procedure 3: Train similarity tree algorithm]
Input: A feedback image and the existing tree model
Output: The updated tree model

1. Set *CurrentNode* = root, *TotalSim* = 0, and the node level i to 1.
2. If all the I features have been learned, then go to step 3.
   (a) Call clustering procedure 4 to obtain clustering center C_{ir} with maximum similarity *MaxSim* to the image's i-th feature.
   (b) Calculate *TotalSim* = *TotalSim*+*MaxSim**u_i. Here u_i is the interfeature weights of the i-th the feature.
   (c) If C_{ir} exists in the children of *CurrentNode*, then set the found child as *CurrentNode*. Go to step (e).
   (d) Add C_{ir} as the new child of *CurrentNode*. Set the new child as *CurrentNode*.
   (e) Set $i = i + 1$. Go to step 2 to process next feature.
3. Label leaf node as a positive (relevant) or negative (irrelevant) clustering pattern.
4. Calculate the total error as: *Error* = *RealSim- TotalSim*. Here *RealSim* is the similarity of the image derived from user feedback.
5. For $i = 1, 2, \ldots, I$, learn the intrafeature weights $\vec{W}_i$ by *Error* using C_{ir} as the query vector of the i-th feature in the adaptive filter Filter i.
6. Learn the interfeature weights $U = [u_1, u_2, \ldots, u_I]$ of the features by *Error* in Filter 0 using the adaptive filtering technique described in Sect. 4.
7. Return the updated tree model.

[Procedure 4: Clustering algorithm for the i-th feature]
Input: An image's i-th *feature and the clustering set* C_i
Output: Cluster C_{ir} *with maximum similarity to the image's* i-th *feature*

1. Calculate the similarity between the image's i-th feature and all clusters of the i-th feature by intrafeature weights $\vec{W}_i$ and find the cluster C_{ir} with the maximum similarity *MaxSim*.
2. If *MaxSim* of C_{ir} is larger than the given threshold, e.g., 0.8, go to step 4.
3. Add the image's i-th feature as the new cluster and set the new cluster as C_{ir}.
4. Update the center of cluster C_{ir} by the image's i-th feature.
5. Return the updated C_{ir} and *MaxSim*.

We can estimate the average uncertainty over all the E_j by

$$E[H_T(E)] = \sum_j P(E_j) H(E_j) \tag{27}$$

Then the average reduction in uncertainty entropy achieved by feature test T is:

$$R_T(E) = H(E) - E[H_T(E)] \tag{28}$$

Select the feature test that gives maximum reduction of uncertainty at the first level and then applies this criterion recursively until all features have been used.

The tree similarity model can learn not only positive samples but also negative samples. After learning feedback sam-

[Procedure 5: Calculate similarity using tree model]
Input: An image and the tree model
Output: Total similarity between them

1. Set *CurrentNode* = root, *TotalSim* = 0, and node level i to 1.
2. If all the I features have been tested, go to step 3.
   (a) Calculate the similarity between *CurrentNode's* children and the image's i-th feature by the intrafeature weights $\vec{W}_i$ of the i-th feature vector and find the child node with the maximum similarity *MaxSim* as the *CurrentNode*.
   (b) If *MaxSim* of the *CurrentNode* is less than a given threshold, e.g., 0.6, go to step 3.
   (c) Calculate current *TotalSim* = *TotalSim* + *MaxSim* * u_i, where u_i is the interfeature weights of the i-th feature.
   (d) Set $i = i + 1$. Go to step 2.
3. If the label of the leaf node in the traced path is a pattern of an irrelevant image, then set *TotalSim* to zero.
4. Return *TotalSim*.

ples, it can cluster all discovered positive and negative samples into organized patterns of the tree. Normally, the positive samples can be just a few patterns for a user's query in CBIR. However, there are many patterns of the negative samples. Although it is difficult to acquire enough negative patterns in a large database by learning a few negative patterns, it can still improve the performance to some extent because learning a few negative samples can reject obvious dissimilar images quickly and can help decide which feature is more important and discriminating. We will provide a comparison between learning by only the positive samples and learning by both positive and negative samples in Sect. 6.6.3.

6 Experiments

In the previous sections, we showed the theoretical advantages of the proposed adaptive tree similarity model and adaptive filter, e.g., optimality, incremental learning, low computation complexity, and ability to model nonlinear Boolean queries. In this section, we will examine the retrieval performance (e.g., accuracy and robustness) via experiments. Specifically, we would like to answer the following questions:

- Which algorithm performs better, LMS, RLS or the existing OPL, MARS, Mind Reader approaches?
- Which sequencing order performs better for adaptive filters, forward or backward and why?
- Which similarity model is better, the linear model or the tree similarity model?
- Which learning process is better, learning using positive samples only or using both positive and negative samples?

6.1 Data set

For all the experiments reported in this section, the Corel image collection is used as the test data set. We choose this data set for the following considerations:

- It is a large-scale data set. Compared with the data sets used in some systems that contain only a few hundred images, the Corel data set includes 17,000 images.
- It is heterogeneous. Unlike the data sets used in some systems that are all texture images or car images, the Corel data set covers a wide variety of content from animals and plants to human society and natural scenery.
- It is professionally annotated by Corel professionals. Instead of using the less meaningful low-level features like the evaluation criterion, the Corel data set has human-annotated ground truth. All the images in their entirety have been classified into 170 categories, and there are 100 images in each category.

The Corel data sets have been used in other systems in which relatively high retrieval performance has been reported [15, 17, 32]. However, those systems use only preselected categories with distinctive visual characteristics (e.g., red cars vs. green mountains). In our experiments, no preselection is made in 17,000 images. Since average users want to retrieve images based on high-level concepts, not low-level visual features [23, 25], the ground truth we use is based on high-level categories such as car, flower, people, etc. In experiments, in order to obtain an objective evaluation of the various retrieval techniques, we use the categories to evaluate the retrieval performance. But in practice, the system is to enable users to guide the system to the images that are meaningful while not being subjected to categorization.

6.2 Queries

Some existing systems only test a few preselected query images. It is not clear if those systems will still perform well on other nonselected images. To fairly evaluate the retrieval performance of different methods and similarity models, we randomly generated 400 queries on all 17,000 images for each retrieval condition. For all the experiments reported in this section, they are the average of all 400 query results.

6.3 Visual features

In our current system, we use three visual features: color moments, wavelet-based texture, and a water-fill edge feature. Every feature has been Gauss normalized beforehand due to different ranges of value. For color moments, we choose to use the HSV color space because of its similarity to human vision perception of color [25]. For each of the three color channels, we extract the first two moments (e.g., mean and standard deviation) and use them as the color feature.

The second visual feature we use is a recently developed water-fill edge feature [40]. The original image is first fed into the Canny edge detector to generate its corresponding edge map. The edge map is then considered as a network of tunnels. Virtual water is then poured into the tunnels. By measuring maximum filling time, maximum fork count, etc., this algorithm captures a total of 18 edge characteristics of the original image.

For the third wavelet-based texture, the original image is fed into a Daubechies-4 wavelet filter bank [6] and decomposed into the third level, resulting in ten decorrelated subbands. Out of the ten sub-bands, nine are "detailed" bands capturing the characteristics of the original image at different

Table 1. Comparison between forward learning (LMS_F) and backward learning (LMS_B) for LMS in the linear model

Precision (percentage)		0 iterations	1 iteration	2 iterations
Return top 20	LMS_F	14.48	16.83	17.50
	LMS_B	14.41	18.96	20.41
Return top 100	LMS_F	6.91	9.94	10.44
	LMS_B	7.25	11.87	13.04
Return top 180	LMS_F	6.18	7.15	9.08
	LMS_B	5.73	9.09	9.58

scales and orientations. The last band is the "smoothed" band, which is a subsampled average image of the original image. For each subband, we extract the standard deviation of the wavelet coefficients and therefore have a texture feature vector of length 10. This wavelet-based feature has been proven to be quite effective in modeling image texture features [25,26].

It is worth noting that the proposed tree similarity model framework is an open framework. More advanced features (e.g., region, semantics, etc. [15,18]) can readily be incorporated into the system for better performance. Here we wish to use the above three visual features to achieve better performance by the proposed learning approaches than the regular techniques.

6.4 Performance measures

The most widely used performance measures for information retrieval are precision (*Pr*) and recall (*Re*) [27]. *Pr* is defined as the number of retrieved relevant objects (i.e., N) over the number of total retrieved objects, say the top 20 images. *Re* is defined as the number of retrieved relevant objects over the total number of relevant objects in the image collection (in the Corel data set case, 99). In general, *Pr* will decrease when *Re* increases. The performance of an "ideal" system should be that *the precision is higher at the same recall value*. Because of this, the *Pr(Re)* curve is used to better characterize the performance of a retrieval system.

6.5 System description

We have developed a prototype system based on the proposed adaptive tree similarity model approaches. The system is written in C++ and runs on the Windows 2000 platform. Its interface is shown in Fig. 5. Our prototype system has two modes: *demo* mode and *testing* mode. During the demo mode, a user can browse through the image collection and submit any image as the query image, which is shown in the top left corner. For each of the returned images, there is a degree-of-relevance (i.e., similarity $\pi(n)$) slider to allow the user to provide his/her relevance feedback. During testing mode, the system randomly selects query images and uses the Corel high-level category label as the ground truth to obtain relevance feedback. This is a very challenging task, and we report detailed experimental results in the next section.

Fig. 5. User interface of the retrieval system

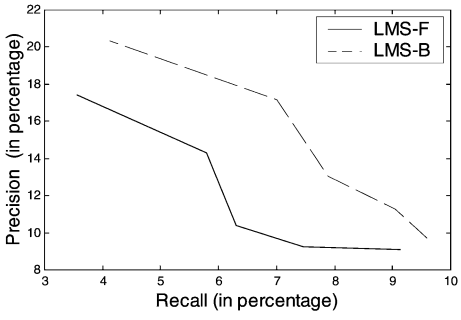

Fig. 6. Comparison between forward learning and backward learning for LMS in the linear similarity model

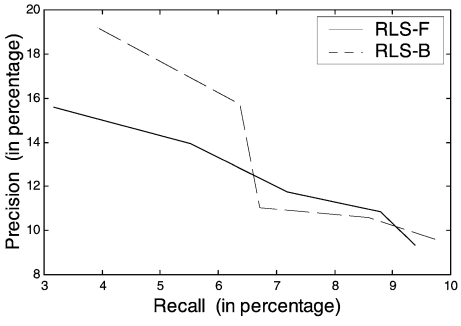

Fig. 7. Comparison between forward learning and backward learning for RLS in the linear similarity model

6.6 Results and observations

6.6.1 Forward learning vs. backward learning

When we use LMS or RLS, we use the linear model as described in Sect. 2. Table 1 shows the forward/backward learning results using a single averaged query example for LMS. When the return top is 20, 100, and 180, most similar images are retrieved. To better compare the two learning orders, in Fig. 6 we also plot the precision-recall curve for LMS after two iterations of feedback. Similarly, Table 2 shows the forward /backward learning results for RLS, and Fig. 7 shows their precision-recall curves.

Regarding forward learning and backward learning, the following observations can be made based on the above tables and figures:

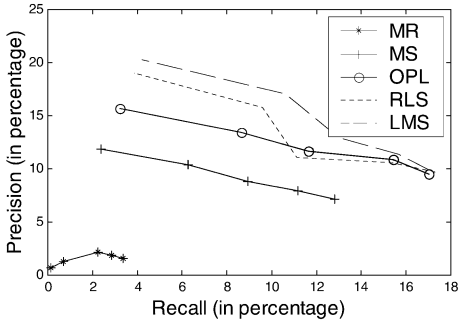

Fig. 8. Comparison between MR, MS, OPL, RLS, and LMS in the linear model

Table 2. Comparison between forward learning (RLS_F) and backward learning (RLS_B) for RLS in the linear model

Precision (percentage)		0 iterations	1 iteration	2 iterations
Return top 20	RLS_F	11.18	12.82	15.62
	RLS_B	14.95	17.86	19.25
Return top 100	RLS_F	7.04	9.68	11.83
	RLS_B	7.23	10.41	11.07
Return top 180	RLS_F	5.89	7.55	9.32
	RLS_B	5.77	9.15	9.63

Table 3. Comparison between Mr, MS, OPL, RLS, and LMS in the linear similarity model model

Precision (percentage)		0 iterations	1 iteration	2 iterations
Return top 20	MR	7.23	0.58	0.29
	MS	7.23	10.99	12.09
	OPL	10.18	14.18	15.85
	RLS_B	14.95	17.86	19.25
	LMS_B	14.41	18.96	20.41
Return top 100	MR	4.36	1.02	2.20
	MS	4.36	7.60	8.82
	OPL	5.75	9.47	11.60
	RLS_B	7.23	10.41	11.07
	LMS_B	7.25	11.87	13.04
Return top 180	MR	3.53	1.06	1.77
	MS	3.53	6.00	7.02
	OPL	4.63	7.78	9.39
	RLS_B	5.77	9.15	9.63
	LMS_B	5.73	9.09	9.58

- For LMS and RLS, the backward learning outperforms the forward learning because the backward learning order simulates the "coarse-to-fine" learning process and benefits the adaptive filtering algorithms to fine-tune the weights at the last stage.
- The ordering effect is more noticeable in LMS than in RLS. When the number of feedback samples is relatively large, the forward and backward learning are almost the same for RLS. This is because RLS continuously learns all the samples, while LMS quickly adapts to the new example, forgetting the older ones.

Table 4. Comparison between LMS in a linear model and TLMSP, TLMSA in the adaptive tree similarity model

Precision (percentage)		0 iterations	1 iteration	2 iterations
Return top 20	LMS_B	14.41	18.96	20.41
	TLMSP	14.05	23.51	26.01
	TLMSA	14.23	24.05	27.96
Return top 100	LMS_B	7.25	11.87	13.04
	TLMSP	7.35	13.54	16.44
	TLMSA	7.28	14.71	18.61
Return top 180	LMS_B	5.73	9.09	9.58
	TLMSP	5.95	11.04	13.53
	TLMSA	5.86	11.71	14.01

6.6.2 LMS vs. RLS vs. OPL vs. MARS vs. MindReader

Table 3 shows the comparison between MindReader (MR) [10], MARS (MS) [26], OPL [24], and LMS with backward learning and RLS with backward learning in a linear model. When the return top is 20, 100, and 180, most similar images are retrieved, and Fig. 8 shows their precision-recall curves after two iterations of relevance feedback. Based on the table and figure, the following observations can be made:

- When the number of returned images is small (e.g., return top 20 images only), the performance of LMS and RLS is significantly better than that of OPL MS and OPL MR. This is because OPL MR and MS get suboptimal feature weights in a linear model when there are not enough feedback samples (see Sect. 3). When the number of returned images is sufficiently large (e.g., 180 images), the performance of the algorithms (LMS, RLS, OPL) is comparable because all three algorithms are close to the optimal solution conditioned on the feedback samples.
- The LMS with backward learning seems to be better than RLS, OPL MS, and OPL MR in a linear model. Not only is its retrieval performance the best, but its computation complexity is also significantly cheaper than that of the other approaches (see Sect. 4.5).

6.6.3 Adaptive tree model vs. linear similarity model

Let TLMSP denote a backward LMS filter to learn only positive samples by the tree model, TLMSA denote a backward LMS filter to learn both positive and negative samples by the tree model, and LMS_B denote a backward LMS filter to learn positive samples by a linear model. They all use the same backward LMS filter to learn but are different in similarity models and learning samples. Table 4 shows the comparison between LMS_B, TLMSP, and TLMSA algorithms. Figure 9 shows their precision-recall curve after two iterations of relevance feedback. The following observations can be made based on Table 4 and Fig. 9:

- TLMSP and TLMSA with the tree similarity model learn much better than LMS_B with the linear model. This is because the tree model is a more accurate nonlinear representation of a user's query concept and is able to find more

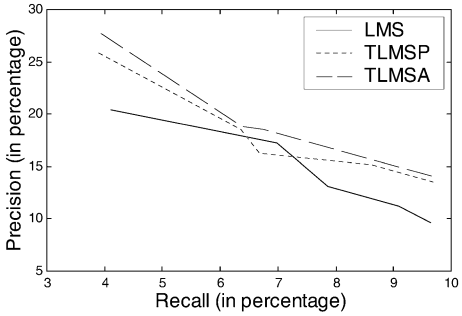

Fig. 9. Comparison between LMS and TLMSP, TLMSA by different similarity models

relevant images by multiple tree paths (subquery patterns). Furthermore, because each path is better approximated as a linear model, the adaptive filter works better in the clustering feature subspace than in the whole nonlinear feature space (e.g., in the LMS_B case).

- TLMSA using both positive and negative feedback samples to learn is better than TLMSP using only positive samples. The reason is that learning negative samples can help the CBIR system to reject many obvious irrelevant images and help the system decide which feature has more discriminatory power than others. However, because there are many patterns for irrelevant images, it is difficult to find all the negative patterns by using just a few training samples. Thus the improvement of TLMSA over TLMSP is not very obvious.

7 Conclusions

Considering the human vision perception system as an intelligent signal filter, we implemented the LMS and RLS adaptive-filter-based feedback technologies to simulate the visual perception model for CBIR. Both LMS and RLS filters learn incrementally and support online adaptive learning with fast convergence and good performance. Furthermore, we extend the conventional linear adaptive filter to a nonlinear adaptive tree similarity model in order to model the complex nonlinear human visual perception. The proposed tree similarity model is a hierarchical nonlinear Boolean representation of clustering patterns of all feedback samples. It decomposes a user's complex query concept into a Boolean combination of multiple simpler subconcepts that spans a smaller feature subspace and can therefore be better learned by the linear adaptive filters. To evaluate the performance of the proposed approach, we conducted extensive tests on a large-scale heterogeneous image collection and compared the results against the state-of-the-art approaches. Experimental results show that the proposed TLMSA approach (using a backward LMS filter to learn both positive and negative samples in the adaptive tree similarity model) is effective and efficient for CBIR.

Acknowledgements. The Corel image set was obtained from Corel and used in accordance with their copyright statement. The authors from Tsinghua are partially supported by the Natural Science Foundation of China (Project Number 69902004), NKBRSF (Project Number 1998030600) and the National Key Project 2001BA201A07.

Appendix A: RLS algorithm for CBIR

For a finite time serial signal $\vec{X}(n)$, we have

$$R_{xx}(n) = \sum_{i=1}^{n} \pi(i)\vec{X}(i)\vec{X}^T(i)$$
$$\vec{P}(n) = \sum_{i=1}^{n} \pi(i)\vec{X}(i)d(i) \tag{29}$$

Hence we have the following recursion for updating the covariance matrix and the crosscorrelation vector

$$R_{xx}(n) = R_{xx}(n-1) + \pi(n)\vec{X}(n)\vec{X}^T(n)$$
$$\vec{P}(n) = \vec{P}(n-1) + \pi(n)\vec{X}(n)d(n) \tag{30}$$

The matrix inversion lemma helps us to obtain the inversion of the matrix R_{xx}. Let A, B, and F all be positive definite matrices; the matrix inversion lemma says [27], if

$$A = B^{-1} + CF^{-1}C^T \tag{31}$$

$$\text{Then } A^{-1} = B - BC(F + C^TBC)^{-1}C^TB \tag{32}$$

Because both $R(n)$ and $R(n-1)$ are positive definite, let

$$A = R(n), \quad B = R^{-1}(n-1),$$
$$C = \vec{X}(n), \quad F = 1/\pi(n)\ . \tag{33}$$

For convenience, let's further define

$$Q(n) = R^{-1}(n),$$
$$e(n) = d(n) - y(n) \tag{34}$$
$$= \sqrt{-2\sigma^2 \ln(\pi(n))} - W^T(n-1)X(n)\ .$$

By substituting Eqs. 33 and 34 into Eq. 32, we have

$$Q(n) = Q(n-1) - \vec{K}(n)\vec{X}^T(n)Q(n-1) \tag{35}$$

where

$$\vec{K}(n) = \frac{Q(n-1)\vec{X}(n)}{[1/\pi(n) + \vec{X}^T(n)Q(n-1)\vec{X}(n)]} \tag{36}$$

is called the gain vector. Rearranging the terms in Eq. 36, we have

$$\vec{K}(n) = Q(n-1)\vec{X}(n) - \vec{K}(n)\vec{X}^T(n)Q(n-1)\vec{X}(n)$$
$$= (Q(n-1) - \vec{K}(n)\vec{X}^T(n)Q(n-1))\vec{X}(n)$$
$$= Q(n)\vec{X}(n) \tag{37}$$

To summarize, the recursive $W(n)$ can be calculated as follows:

$$\vec{W}(n) = R_{xx}^{-1}(n)\vec{P}(n)$$
$$= Q(n)\vec{P}(n) \ = Q(n)[\vec{P}(n-1) + \vec{X}(n)d(n)]$$
$$= Q(n-1)\vec{P}(n-1)$$
$$-\vec{K}(n)\vec{X}^T(n)Q(n-1)\vec{P}(n-1)$$
$$+Q(n)\vec{X}(n)d(n)$$
$$= \vec{W}(n-1) + \vec{K}(n)(d(n) - \vec{X}^T(n)\vec{W}(n-1))$$
$$= \vec{W}(n-1) + \vec{K}(n)e(n)$$

References

1. Bergen J, Adelson EH (1987) Hierarchical, computationally efficient motion estimation algorithm. J Optical Soc Am pp 4–35
2. Bradshaw B (2000) Semantic based image retrieval: a probabilistic approach. Proceedings of the 8th ACM multimedia conference, Los Angeles, 30 October– 4 November 2000, pp 167–177
3. Fox E, Marchionini G (guest eds) (2001) Digital libraries: introduction. Commun ACM (special issue on digital library) 44(5):30–32
4. Cox IJ, Miller ML, Omohundro SM, Yianilos PN (1996) PicHunter: Bayesian relevance feedback for image retrieval. Proceedings of the international conference on pattern recognition, Vienna, 30 October–4 November 1996, vol 3, pp 361–369
5. Cox IJ, Miller ML, Minka TP, Papathomas TV, Yianilos PN (2000) The Bayesian image retrieval system, Pichunter: theory, implementation, and psychophysical experiments. IEEE Trans Image Process 9(1):20–37
6. Daubechies I (1992) Ten lectures on wavelets, CBMS–NSF lecture notes 61, SIAM
7. Dimitrova N, Sethi I, Rui Y (2000) Media content management. In: Syed MR (ed) Design and management of multimedia information systems: opportunities and challenges. Idea Publishing Group, Hershey, PA
8. Gudivada VN, Raghavan VV (guest eds) (1995) Content based image retrieval systems. IEEE Comput Mag (special issue on content-based image retrieval systems) 28(9):18–22
9. Haykin S (1996) Adaptive filter theory, 3rd edn. Prentice-Hall information and system sciences series. Prentice-Hall, Upper Saddle River, NJ
10. Ishikawa Y, Subramanya R, Faloutsos C (1998) Mindreader: query databases through multiple examples. Proceedings of the 24th VLDB conference, New York, 24–27 August 1998, pp 651–675
11. Jain R, Murthy S, Chen P (1995) Similarity measures for image databases, Proceedings of the IEEE conference on fuzzy systems, Yokohama, 20–24 March 1995, vol 3, pp 1247–1254
12. Krumhansl CL (1978) Concerning the applicability of geometric models to similarity data: the interrelationship between similarity and spatial density. Psychol Rev 85:445–464
13. Lee C, Ma WY, Zhang HJ (1998) Information embedding based on user's relevance feedback for image retrieval. Technical report HP Labs
14. Lee HK, Yoo SI (2000) A neural network-based image retrieval using nonlinear combination of heterogeneous features. Proceedings of the 2000 congress on evolutionary computation, LA Jolla, CA, 16–19 July 2000, vol 1, pp 667–674
15. Li J, Wang J, Wiederhold G (2000) IRM: integrated region matching for image retrieval. Proceedings of the 8th ACM multimedia conference, Los Angeles, 30 October–4 November 2000, pp 147–157
16. Liu WY, Wang T, Zhang HJ (2000) A hierarchical characterization scheme for image retrieval. Proceedings of the IEEE international conference on image processing (ICIP'2000), vol 3, Vancouver, BC, Canada, 10–13 September 2000, pp 42–45
17. Lu Y, Hu CH, Zhu XQ et al (2000) A unified framework for semantics and feature based relevance feedback in image retrieval system. Proceedings of the 8th ACM multimedia conference, Los Angeles, 30 October–4 November 2000, pp 31–39
18. Ma WY, Zhang HJ (1998) Benchmarking of image features for content-based retrieval. Proceedings of the 32nd Asilomar conference on signals, systems and computers, Monterey, CA, 1–4 November 1998, vol 1, pp 253–257
19. MacArthur SD, Brodley CE, Shyu C (2000) Relevance feedback decision trees in content-based image retrieval. Proceedings of the IEEE workshop on content-based access of image and video libraries, Hilton Head Island, SC 16 June 2000, pp 68–72
20. Minka T, Picard R (1997) Interactive learning using a society of models. Proceedings of the IEEE international conference on computer vision and pattern recognition, San Francisco, pp 447–452
21. Ortega M, Rui Y, Chakrabarti K et al (1998) Supporting ranked Boolean similarity queries in MARS. IEEE Trans Knowledge Data Eng 10(6):905–925
22. Peng J (2000) Adaptive multi-class metric content-based image retrieval. Proceedings of the 4th international conference on visual information systems, Lyon, France, November 2000
23. Rodden K, Basalaj W, Sinclair D, Wood K (2000) Does organization by similarity assist image browsing. Proceedings of the ACM conference on human factors in computing systems, The Hague, 1–6 April 2000, pp 190–197
24. Rui Y, Huang T (2000) Optimizing learning in image retrieval. Proceedings of the IEEE international conference on computer vision and pattern recognition, Hilton Head Island, SC, 13–15 June 2000, vol 1, pp 236–243
25. Rui Y, Huang T, Chang SF (1999) Image retrieval: current techniques, promising directions and open issues. J Visual Commun Image Represent 10:39–62
26. Rui Y, Huang T, Mehrotra S (1997) Content-based image retrieval with relevance feedback in MARS. Proceedings of the IEEE international conference on image processing, Santa Barbara, 26–29 October 1997, pp 815–818
27. Salton G, McGill MJ (1982) Introduction to modern information retrieval. McGraw-Hill, New York
28. Smeulders AWM, Worring M, Santini S et al (2000) Content-based image retrieval at the end of the early years. IEEE Trans PAMI 22(12):1349–1380
29. Swets DL, Weng JJ (1996) Using discriminant eigen-features for image retrieval. IEEE Trans PAMI 18(8):831–836
30. Tian Q, Hong P, Huang TS (2000) Update relevant image weights for content-based image retrieval using support vector machines. Proceedings of the IEEE international conference on multimedia and expo, New York, 31 July–2 August 2000, vol 2, pp 1199–1202
31. Tieu K, Viola P (2000) Boosting image retrieval. Proceedings of the IEEE international conference on computer vision and pattern recognition, Hilton Head Island, SC, 13–15 June 2000, vol 1, pp 228–235
32. Tong S, Chang E (2001) Support vector machine active learning for image retrieval. Proceedings of the 9th ACM multimedia conference, Ottawa, Canada, 30 September–5 October 2001, pp 107–118
33. Trunk GV (1979) A prbolem of dimensionality: a simple example. IEEE Trans PAMI 1:306–307
34. Vasconcelos N, Lippman A (2000) A probabilistic architecture for content-based image retrieval. Proceedings of the IEEE international conference on computer vision and pattern recognition, Hilton Head Island, SC, 13–15 June 2000, vol 1, pp 216–221
35. Wang T, Rui Y, Hu SM (2001) Optimal adaptive learning for image retrieval. Proceedings of the IEEE international conference on computer vision and pattern recognition, Kauai, Hawaii, 9–14 December 2001, pp 1140–1147
36. Widrow B, Hoff ME (1960) Adaptive switching circuits. Proceedings of the IRE western electronic show and convention, New York, 23 August 1960, pp 96–104
37. Woo TK (2001) HRLS: a more efficient RLS algorithm for adaptive FIR filtering IEEE Commun Lett 5(3):81–84

38. Wu Y, Tian Q, Huang TS (2000) Discriminant-EM algorithm with application to image retrieval. Proceedings of the IEEE international conference on computer vision and pattern recognition, Hilton Head Island, SC, 13–15 June 2000, vol 1, pp 222–227
39. Zhou X, Hunag TS (2001) Comparing discriminate transformations and SVM for learning during multimedia retrieval. Proceedings of the 9th ACM Multimedia Conference, Ottawa, Canada, 30 September–5 October 2001, pp 137–146
40. Zhou X, Rui Y, Huang TS (1999) Water-filling algorithm: a novel way for image feature extraction based on edge maps. Proceedings of the IEEE international conference on image processing, Kobe, Japan, 24–28 October 1999, pp 570–574

Tao Wang is a PhD. student in the Department of Computer Science and Technology at Tsinghua University. He received his B.S. from the University of Science and Technology in China in 1996 and his M.S. from the Chinese Academy of Science in 1999. His current research intersets are in content-based image retrieval, pattern recognition, image processing, and computer graphics.

Yong Rui received his Ph.D. in electrical and computer engineering in 1999 from the University of Illinois at Urbana-Champaign. Since March 1999 he has been a researcher at Microsoft Research in Redmond, WA. His research interests include multimedia systems, distance learning, distributed meetings, image/video/audio processing, computer vision, and machine learning. He has published five book chapters, six journal papers, and over 40 refereed conference papers in the above areas. Dr. Rui holds six U.S. patents pending and is a member of ACM and IEEE.

Shi-Min Hu is an associate professor in the Department of Computer Science and Technology at Tsinghua University in China. He received his B.S. in mathematics from Jilin university in China in 1990, his M.S. and Ph.D. in computer aided geometric design and computer graphics from Zhejiang university in China in 1993 and 1996, and he finished his postdoctoral research at Tsinghua University in 1998. His current research intersets are in computer-aided geometric design, computer graphics, and content based image retrieval.

Jia-Guang Sun is a professor in the Department of Computer Science and Technology at Tsinghua University. He is also an Academician in the Chinese Academy of Engineering and the Director of the National CAD Engineering Center at Tsinghua University. He received his B.S. in Computer Science from Tsinghua University in 1972. From 1985 to 1986, he was a visiting scholar at UCLA. His current research intersets are in computer-aided geometric design, computer graphics, and product data management.