Computer Vision and Image Understanding **88,** 55–75 (2002)
doi:10.1006/cviu.2002.0977

Interactive Content-Based Image Retrieval Using Relevance Feedback

Sean D. MacArthur, Carla E. Brodley, and Avinash C. Kak

School of Electrical and Computer Engineering, Purdue University, 1285 Electrical Engineering Building, West Lafayette, Indiana 47906
E-mail: macarthu@ecn.purdue.edu, brodley@ecn.purdue.edu, kak@ecn.purdue.edu

and

Lynn S. Broderick

Department of Radiology, University of Wisconsin-Madison, Madison, Wisconsin 53706
E-mail: lsbroderick@facstaff.wisc.edu

Received August 17, 2001; accepted October 29, 2002

Database search engines are generally used in a one-shot fashion in which a user provides query information to the system and, in return, the system provides a number of database instances to the user. A relevance feedback system allows the user to indicate to the system which of these instances are desirable, or relevant, and which are not. Based on this feedback, the system modifies its retrieval mechanism in an attempt to return a more desirable instance set to the user. In this paper, we present a relevance feedback technique that uses decision trees to learn a common thread among instances marked relevant. We apply our technique in a preexisting content-based image retrieval (CBIR) system that is used to access high resolution computed tomographic images of the human lung. We compare our approach to a commonly used relevance feedback technique for CBIR, which modifies the weights of a K nearest neighbor retriever. The results show that our approach achieves better retrieval as measured in off-line experiments and as judged by a radiologist who is a lung specialist.

Key Words: content-based image retrieval; relevance feedback; decision trees.

1. INTRODUCTION

Content-based image retrieval, or CBIR, systems have gained popularity because of their objective means of assessing image content. Textual annotation, in contrast, is plagued by

1077-3142/02 $35.00

inconsistencies of the human annotator. Perhaps more important, in many domains, text cannot accurately capture the visual attributes of an image in the user's mind. In CBIR, the query is an image that the user presents as an example of what he or she is looking for. The goal of CBIR is to retrieve images that are visually similar to the query image. To this end, a CBIR system applies image processing and computer vision algorithms to extract a vector of features from each of the database images. Typical features might include pixel color histograms, gray scale histograms, texture features, and edge-content measures. If the feature vectors are composed of n elements, each vector can be interpreted as a point in n-dimensional space. CBIR systems rest on the assumption that points in this space that are proximal to the point represented by the query image's vector should correspond to the feature vectors of images that are visually similar to the query. Similarity to the query is computed using either a default or a user-defined similarity metric. For example, in the popular k nearest neighbor retrieval method, the vectors with the minimal Euclidean distance to the query vector are retrieved. (Note that some type of normalization of feature scales is required prior to this step.)

In the context of image retrieval, an unweighted nearest neighbor retriever has at least one drawback: it assumes that all features are equally relevant [10]. Some features may be meaningless to the user or may be highly correlated with other features. Some features may be very useful for certain queries but may lose significance for other queries [3]. The notion of "similar" in the mind of the user may fluctuate depending on the query, the history of retrievals observed, and the user. If there is a significant discrepancy between the similarity as calculated by the system and the notion of similarity in the user's mind, the results are destined to be unsatisfactory [10]. This problem has served as the impetus for what is known as *relevance feedback*.

Relevance feedback retrieval systems prompt the user for feedback on retrieval results and then use this feedback on subsequent retrievals with the goal of increasing retrieval performance. A typical user-system session is as follows: A user presents an image query to the system whereupon the system retrieves a fixed number of images using a default similarity metric. The user then rates each returned result with respect to how useful the result is for his or her retrieval task at hand. Ratings may be simply "relevant" or "not relevant" or may have finer gradations of relevancy such as "somewhat relevant," "not sure," and "somewhat irrelevant." The relevance feedback algorithm uses this feedback information to select another set of images to retrieve for the user; whether the new and previous sets are disjoint depends on the particular system. The system's goal is to effectively infer which images in the database are of interest to the user based on this feedback. The user could then rate these images in the second set in a similar way and the process may iterate indefinitely in this closed-loop fashion.

A relevance feedback retrieval system has a number of design requirements that allow the system to function in an efficient online manner.

- After each iteration, when a set of images is retrieved, the system must require a reasonable amount of feedback. If the user needs to labor over providing feedback for numerous images after each iteration, they will tire quickly and not be satisfied with the process.
- The system must produce acceptable results after only a few iterations. If large numbers of iterations are required, the user will also tire.
- Feature extraction should be completed in a short period of time to prevent user frustration.

Reasonable query feature extraction time, feedback per iteration, and number of iterations are a necessity. Exact bounds on what constitutes "reasonable" are domain and user specific, but rough bounds are determined by common sense (e.g., rating four images is reasonable, while rating 40 is not).

In this paper, we present a new relevance feedback retrieval system that uses machine learning to infer which images in the database would be of most interest to the user at a particular point in time. Specifically, decision trees are constructed to discover a "common thread" among those images marked relevant. An empirical evaluation demonstrates (a) how retrieval performance increases significantly after only one or two iterations, (b) that the user need only provide feedback on a handful of images, and (c) that the system can be used in real time.

This paper is divided into the following remaining sections. Section 2 describes existing work related to relevance feedback in CBIR. Section 3 delves into the intricacies of our retrieval system at an algorithmic level. In Section 4, using a medical image database, we compare our system's performance to that of another recent relevance feedback retrieval system. Finally, Section 5 summarizes the contributions made in this paper.

2. RELATED WORK

The majority of relevance feedback methods for CBIR employ a weighted nearest neighbor retrieval mechanism. This practice has a pleasant interpretation, because a larger weight on a feature intuitively signifies that feature has greater relevance. In MARS, Multimedia Analysis and Retrieval System, [12], a feature's weight is determined by examining the feature's variance across the set of retrieved images marked as relevant by the user. A low variance indicates that these relevant images are consistent in this feature and that the feature should be assigned a relatively high weight. A feature whose value across the relevant images varies significantly is, conversely, given a relatively small weight. Thus, a feature's weight is assigned in inverse proportion to the feature's variance across the images marked relevant. Documented results of this technique involve experiments where the number of marked images exceeds 25. It is unclear how this technique would fare given only a handful of marked images.

The relevance feedback problem has been cast as a vector equation optimization problem [14]. This is a natural step given the matrix equations describing a weighted distance function. One goal is to move the query to a position in the feature space away from negative example instances and toward positive example instances. The procedure involves two steps: (1) finding the optimal query position and (2) finding the optimal feature weighting for the new query to maximize precision. The system of equations is constrained, but is transformed into an unconstrained one through the use of Lagrange multipliers. The solution is then converted back into a solution for the original problem. This attempts to find an optimal weighting for the weighted distance function, but any such weighting can only distort the feature space in the directions of the space's axes. It cannot, for example, retrieve efficiently from a bimodal distribution of relevant instances with respect to a useful feature. With this method, learning multiple, disjoint intervals of relevancy with respect to a feature is impossible.

A simple weighted distance function is more general than one using uniform weights, but it still cannot capture interactions between features. With uniform weights, the distributions

of the features are assumed to be independent. In [4], a relevance feedback system is presented that can learn diagonal queries, that is, queries where two or more features have nonzero covariance. The system is able to learn generalized ellipsoids by approximating covariance matrices to fit the training instances as well as providing a mechanism for movement of the query position. Continuing in this vein, Seidl and Kriegel [15] present a similar framework for learning generalized distance functions such as ellipsoids. In particular, they have adapted and elaborated upon much of this theory in terms of color. Color histograms, with up to hundreds of constituent bins, may be manipulated within this generalized learning framework. Learning generalized ellipsoids improves upon systems such as MARS, but such a framework still cannot learn bimodal distributions of relevant instances along a feature.

The PFRL (Probabilistic Feature Relevance Learning) retriever [10] employs a weighted nearest neighbor method, but the technique used to map user feedback to feature weights is somewhat different. A feature's weight is computed by examining the C marked images closest to the query with respect to only that feature. The higher the frequency of images marked relevant within this set of images, the higher the weight that is assigned to that feature. Retrieval precision for this technique has also only been documented for a large number of images marked per iteration [10]. In Section 4, we illustrate empirically some of the weaknesses of this method for the domain of high resolution computed tomography (HRCT) of the lung.

Typically, categorical relevance feedback is provided to a system: an instance may be marked as either relevant or irrelevant. In [2], Cox *et al.* recognize that this places an added burden on the user. The user must attempt to not delineate too many instances as either relevant or irrelevant—if the positive (negative) feedback is too plentiful, the query will be too broad (narrow), resulting in less than satisfactory results. Instead, the authors present a system where comparative feedback is supplied such as, "instance A is more relevant than instance B." The problem is then to locate an ideal query that maximally satisfies these comparative constraints. Large sections of the feature space can be removed from consideration, which makes the task computationally efficient. Furthermore, database instances may be categorized using a *vantage point tree* (similar to a k-d tree) that partitions the feature space greedily into sections described by the comparative constraints. In addition, provisions exist for making the trees *soft*, whereupon they can index data using probabilities of constraints being valid instead of hard constraints. Unfortunately, a user may waste time and effort deciding relative relevance among retrievals that, perhaps, should have all been rated irrelevant.

More recently, support vector machines, or SVMs, have been employed for learning an effective retrieval strategy. SVMs nonlinearly transform feature vectors into a space such that the vectors marked relevant can be enclosed in a hypersphere in the new space. By this design, points that are most interior to the hypersphere should be most relevant. Chen *et al.* [1] describe a unique way to control expansion and contraction of the hyperplane boundary via a parameterized energy function that effectively trades off overfitting and overgeneralization. Our method, however, selects feature space regions to investigate without the need for parameters. The SVM framework is further exploited in [22]. Tong and Chang use active learning to select images to present to the user that are near the current hyperplane boundary in order to better refine the boundary. This continues until the final iteration, at which point the images that are most interior to the hyperplane are returned. A drawback of this type of system is that the user has to toggle between marking marginal images and retrieving the best images. Our system balances query refinement and retrieval of quality images seamlessly.

In [25], Zhou and Huang use biased descriminant analysis to address the problem that while relevant retrievals have common attributes, irrelevant retrievals may be different from the positive examples in any number of ways. The research proposes a transform to cluster relevant examples in a new space while avoiding efforts to cluster the negative examples. It cites this assymetric treatment of the two classes as beneficial to retrieval performance.

Tieu and Viola [21] apply boosting, a technique that combines numerous weak classifiers to yield a strong one, to learn to efficiently browse an image database. Thousands of low level visual features are extracted from the database images. Only 20 have their weights boosted to the degree that they are used by a classifier. The user selects a few examples of images that he or she is interested in and the system selects 100 negative examples randomly. Boosting is applied and the resulting classifier is used to generate more examples for the user to examine. The process repeats until the user is satisfied. It is unclear whether a single positive example (a query image in our domain) would be sufficient to learn the underlying concept.

In [6], images in the database are characterized by their color histograms. Each continuous color frequency bin may be discretized into intervals, thus converting the histograms into discrete feature vectors. The discretization is governed by a mechanism to minimize the class entropy in each resultant interval. Weights are assigned to each interval; the greater the discriminating power, the greater the weight. Initially, a weighted k-nearest neighbor retrieval is performed that returns a ranked image list. A decision tree is then induced whose leaves contain a measure of the confidence that a feature space region contains relevant images. The retrieved images are filtered through the tree and each is then reranked by the weight in the leaf it is routed to. In contrast, what we will demonstrate is that our method first induces a decision tree to pool images to be retrieved. We then execute an unweighted k-nearest neighbor retrieval on the pool using a subset of the available features to return a ranked image list. We briefly include these details here to delineate differences between the system in [6] and our own.

3. RELEVANCE FEEDBACK DECISION TREES

We view the task of relevance feedback as a machine learning problem. The view of this problem as a two class classification problem was first suggested by van Rijsbergen [23]. Our algorithm, named RFDT (relevance feedback decision tree retriever), endeavors to classify an arbitrary image into one of two classes: relevant to the user's needs or not.

At a high level, the system operates as follows: The user provides a query image, which is automatically labeled relevant because it is the standard of relevance against which other images in the database will be compared. A nearest neighbor retrieval is invoked to retrieve the first set of K images—the standard procedure in the absence of any feedback. The user then marks these K retrieved images as either relevant or irrelevant and these markings, along with the query (marked relevant), are relayed back to the system as feedback. For each of the K retrieved images, its relevancy class marking together with its associated feature vector form the training data. This set is used to train a decision tree to distinguish relevant from irrelevant images. Next, each database image is classified into one of these two classes by the tree. Those images that are classified as relevant are pooled together and the K of them closest to the query in Euclidean distance are the next set to be returned to the user for perusal. Once again, the user can mark these images providing the system with

$2K + 1$ feature vectors with relevancy markings to comprise the next decision tree training set. The full history—the query plus both user-marked sets—is used. Each tree and thus each retrieval gets more and more refined with accumulating feedback information for a particular query. This process can iterate indefinitely until the user is satisfied.

3.1. An Overview of Decision Trees

A decision tree is a method for recursively partitioning a feature space containing data instances such that, at termination, each partition is populated by instances of single class value. See [11] for an in-depth presentation. One criterion for making these sequential cuts in the space orthogonal to the feature axes is based on an information-theoretic concept called *entropy* [16]. The entropy of a given partitioning of the space is a measure of the heterogeneity of the class distribution across the partitions. Thus, partitions that are more purely of one class have a lower entropy. A cut is selected to maximize the cumulative decrease in entropy after the cut has been made. Thus, cuts are added in order to increase the class purity of the resultant partitions. The partitioning stops when each of the partitions is populated by instances of a pure class.

For many datasets, after a tree is grown that is consistent with the training data, a pruning procedure is run to avoid overfitting to the data. A pruned tree will be less consistent with the specific training data in order to better generalize to other data. We do not prune because we require the tree to be completely consistent with the training data, which will be elaborated upon later in this section.

This recursive partitioning can be expressed as a tree data structure. The root of the tree represents the entire space; if it has children, they each represent the space after being partitioned by the first cut. The feature index and the value of the feature at the cutting plane are recorded in the root. Similarly, each of the child nodes represents the two subset spaces and each can store similar feature information if the corresponding subset spaces are further subdivided. A leaf node represents one of the final, class-pure partitions; it is labeled with the class of the data in the partition.

Once a tree has been constructed (or *induced*), it serves as a predictive model. The tree uniquely defines a partitioning of the space, with each partition having an associated class. An unclassified instance can be projected to this partitioned space; it is then labeled with the class associated with the partition.

3.2. The RFDT Algorithm

On the first iteration, no feedback information has been provided, so the retriever performs an unweighted K nearest neighbor retrieval using the feature vector of the query and those of the images populating the database. (Note that the algorithm may be configured to use all of the features in this calculation, or, alternatively, may only use a subset of the complete feature set. The benefits of the latter choice are described later in this section.) From the database, K images are retrieved and displayed for the user. The user then marks the retrieved images as relevant or irrelevant as he or she sees fit. These feature vectors are relayed back to the system and the second iteration begins. On the second iteration, the algorithm induces a decision tree from the $K + 1$ labeled feature vectors by using C4.5 [11]. This induction may use the same feature subset used in the initial retrieval, or it may use all of the features. We evaluate these choices in Section 4. Figure 1a provides an example

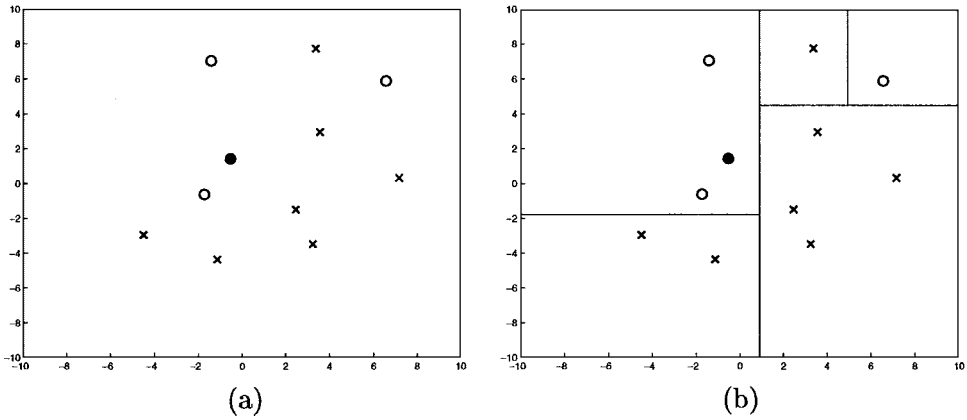

FIG. 1. (a) Feedback vector data and (b) the corresponding partitions found by a decision tree.

of two-dimensional feature vector data, where each point represents a user-labeled feature vector corresponding to an image. The open circles represent vectors marked relevant, the x's represent vectors marked irrelevant, and the solid circle represents the query. Figure 1b illustrates the same space after a decision tree inducer has determined partitions in the feature space. Note that partitions have complete class purity. (The query is in the same partition as vectors marked relevant. This does not violate this purity stipulation because the query is relevant by default.) The C4.5 routine is executed with its default options, except that we allow leaf nodes to have a minimum of one training instance. This is accomplished by using the "$-m$ 1" option in C4.5.

Once the tree is formed, we use it to select the next set of K images to present to the user. To this end, we classify the entire database of feature vectors via the learned tree. That is, the tree is used to route each database image's feature vector down to a leaf node. This is equivalent to projecting the instance into the partitioned space. When an image is routed to a particular leaf, the unique index of the image in the database is stored in that leaf. Thus, a leaf has a record of all of the images that have been routed to it. (We will say that images that have been routed to a node are *in* that node.) When all instances in the database have been filtered through the decision tree, the instances contained in the leaves labeled with class "relevant" are assembled into a list. From this list, the K images closest to the query are retrieved by executing an unweighted K nearest neighbor retrieval on only this list. These K are then presented to the user for another round of relevancy marking. On the next iteration, the retriever's operation is identical to that of the second iteration, except that now there are a total of $2K$ instances labeled with user feedback, plus the query, from which the system will induce a new decision tree. Thus, each subsequent iteration allows the retriever to learn from K more images than the previous iteration, as long as database images are retrieved for the user without replacement. All retrieval experimentation, barring tests of the system's earliest incarnation, was performed without replacement; that is, each image may be retrieved once and only once. (Replacement can both artificially inflate popular performance metrics as well as frustrate the user with remarkings.) This retrieve–feedback loop continues until the user becomes satisfied with the result or until the user feels that the process has exhausted his or her resources for finding relevant images in the database.

There are contingencies for certain scenarios that may arise in the application of the decision tree. If the list of all the images that are in relevant nodes of the tree has fewer

than K items, this indicates that the tree is overly specific, or *overfitted*, to the training data. The predictive model must be "relaxed" in a way as to allow more images to be classified as relevant. This is based on the premise that unseen relevant images always exist in the database. More complex trees describe more specific predictive models, so the system makes the tree less specific by *pruning*. Pruning, in this context, collapses the deepest pair of sibling nodes. They are removed from the tree and their common parent node becomes a leaf node *labeled as relevant*. Also, the two image sets that were associated with the siblings are combined to form the image set of the parent. Note that one of the now deceased nodes was labeled irrelevant; this node's images now populate the parent node and are considered relevant. Thus, the overall number of images classified as relevant must necessarily increase. At this point, the system again counts the number of images marked relevant in the tree. If it still less than K, the pruning process is repeated until K or more are available.

Pruning can be executed in a number of ways. The pruning method employed in our method prunes the deepest sibling node pair. (Strictly speaking, if another node pair exists at the maximal depth, it may be pruned instead.) The frequency with which pruning must be executed depends on the dataset and happens more frequently as the number of iterations grows. After 10 iterations pruning occurred only in 10% of retrievals. Instead of pruning the deepest node pair, a system could prune the node pair with the fewest number of images in them or even attempt all possible pruning operations and evaluate each of them by some criterion. In general, different pruning methods will excel over other methods depending on the data. It is not our goal to tailor any part of the system to the data we have in any way. To truly evaluate which pruning method is best in a practical sense would require experimentation across numerous large datasets and is beyond the scope of this research.

It may be the case that the tree consists of only one node, a condition that occurs when the tree is pruned down to the root, which will be marked relevant. Note that in this case, the algorithm was not able to partition the space based on the feedback data in such a way that K images fell into relevant partitions. The single node must be marked relevant because its children were just pruned; thus, all remaining instances in the database are members of the relevant instance pool. At this point, the algorithm will simply return the next-nearest K images to the user. Without a tree to differentiate relevant from irrelevant, the system explores outward in all directions by degenerating into a nearest neighbor retriever.

3.3. Configurations Using Feature Subsets

Using all the features for a given retrieval may provide less than optimal results because some of the features may be noisy, irrelevant, or highly correlated with other features. The features used by the initial retrieval play the important role of dictating which locales of the feature space are explored first. Database instances have differing proximities to the query in different feature subspaces. The features used to construct the RFDT affect where in the database the system will focus its next retrieval. In the remainder of this section, we first describe two criteria for selecting a feature subset with which the RFDT retriever will operate. We then describe how we activate different subsets during different iterations for a query in order to increase retrieval performance.

We have investigated the following two feature subset selection criteria. They are designed to work in conjunction with a sequential forward selection (SFS) search [5] wrapped around the nearest neighbor. We use nearest neighbor because this is the method used by the system for retrieval. SFS begins with an empty set of features. It then adds a prospective feature

and performs a set of K nearest neighbor retrievals using each of the instances as the query once. For each retrieval, it calculates a criterion value, explained below. The average of these is stored and associated with the feature. This process iterates, each time using a different prospective feature. The feature whose criterion value average is greatest is permanently added. The entire algorithm restarts, but now has one feature instead of none. Features are selected in this way until no feature can be added to increase the previous round's best average criterion value.

The best mean accuracy criterion. In browsing databases whose constituents have class labels, the user may wish to see instances from the database whose classes match that of the query. This requires a feature set selected such that a nearest neighbor retrieval using it returns, on average, the greatest number of database instances whose class matches that of the query. In SFS, we can evaluate a candidate subset using this criterion by performing a one nearest neighbor leave-one-out cross-validated retrieval. In such a scenario, each database instance is used as a query exactly once as outlined above; if its nearest neighbor (in the candidate feature subspace) matches its class we tally a one, otherwise we tally a zero. Eight features, denoted Subset A, were selected by this technique for our dataset. (Details of our dataset are in the following section). The features chosen are shown in Table 3, along with the criterion value for the chosen subset. In Table 3 we show the criterion value for each class. The bias of SFS toward favoring the majority class is evident from this table. Indeed, the selected feature subset allows retrieval of 80% precision of the majority class, but achieves 0% precision for classes with 27 or fewer images.

TABLE 1
Features and Associated Indices

Feature name	Indices	Feature name	Indices
global mean	0	pbr homogeneity 1	74–78
global std	1	pbr homogeneity 2	79–83
global area	2	pbr contrast	84–88
global histogram	3–18	pbr correlation	89–93
global gray min max	19–20	pbr cluster	94–98
global otsu threshold	21	pbr edge	99–106
fissure length	22–23	pbr global gray diff	107–110
fissure centroid	24–25	pbr seg area mean std max	111–113
fissure endpoint	26–29	pbr seg area hist	114–121
fissureorientation	30	pbr seg gray hist	122–137
pbr centroid	31–32	pbr fissure valid	138–140
pbr area	33–34	pbr closest fissure ID	141
pbr gray minmax	35–36	pbr deviation from fissure	142–143
pbr axes	37–38	pbr distance from fissure	144–145
pbr covariancematrix	39–42	pbr orientation from fissure	146
pbr histogram	43–58	pbr bronchial structure	147–153
pbr mean	59	pbr nodules	154–167
pbr std	60	pbr calcification	168–171
pbr orientation	61	pbr honeycombing	172–174
pbr dist from boundary	62–63	pbr small cystic	175–178
pbr energy	64–68	pbr big cystic	179–183
pbr entropy	69–73		

TABLE 2
Subset Selection Criterion Values per Class

Class index	Disease name	Class frequency	Criterion A (%)	Criterion B (%)
1	Centrilobular emphysema	654	74.7	80.9
2	Hemorrage	383	43.2	55.6
3	Bronchiectasis	234	59.2	57.7
4	MCTD	135	30.9	58.5
5	Boop	74	5.7	18.9
6	LIP	64	25.0	34.4
7	Lymphangitic carcinomatosis	61	0.0	0.0
8	IPF	58	31.0	43.1
9	Edema	57	0.0	75.4
10	Cyst	37	37.2	21.6
11	Aspergillus	34	8.1	20.6
12	MAI	27	0.0	0.0
13	Alveolar proteinosis	17	0.0	0.0
14	Bronchilitis obliterans	15	0.0	0.0
15	Coccidioidomycosis	15	0.0	0.0
16	Metastatic calcification	15	0.0	0.0
17	Honeycombing	10	0.0	0.0
18	Hypersensitivity pneumonitis	8	0.0	0.0
19	Histoplasmosis	6	0.0	0.0
20	DIP	5	0.0	0.0
21	Asbestosis	4	0.0	0.0
22	Aspergilloma	3	0.0	0.0

The at-least-one-match criterion. For some queries, a nearest neighbor retrieval may not return any database instances whose class matches that of the query, leaving the user with no relevant images. Our second feature subset selection criterion tries to maximize the likelihood that at least one of the n retrieved instances has the same class as the query. If the user can see as few as one retrieved instance that is relevant, it may give him or her a foothold for training the RFDT retriever to explore the relevant part of the feature space. Here, SFS operates exactly like in the best class accuracy case with two differences. First, it is wrapped around a four nearest neighbor retriever. Second, the tally calculated per instance is a 1 if one or more retrieved instances matches the class of the current instance. As in the previous case, these values are summed and divided by the total number of instances to determine a criterion value for a particular subset. Sixteen features, denoted Subset B, were selected by this technique for our data. Table 2 shows the mean criterion values per class, and Table 3 shows the mean criterion values across all classes.

TABLE 3
Feature Subsets and Selection Criteria

Subset name	Criterion type	Criterion value	Resultant subset
Subset A	Best mean accuracy	46.4%	2, 3, 6, 8, 10, 12, 13, 20
Subset B	At-least-one match	56.1%	0, 3, 4, 5, 7, 8, 9, 13, 14, 19, 21, 22, 62, 105, 187, 198

3.4. Subsets Used in Our Experiments

In its current incarnation, we allow the algorithm three modes of operation. First, we allow all of the features to be used in both the initial K nearest neighbor retrieval and all decision tree inductions on later iterations. As an alternative, we choose one of the two subsets described above to be used for both the initial retrieval and decision tree induction during all of the subsequent retrievals. As a third option, we select a feature subset to be used only by the initial k nearest neighbor retrieval. All subsequent retrievals performed for a given query in this third mode use decision trees induced from all of the features and retrieve from the relevant instance pool using all features as well. This hybridized technique is motivated by the notion that the initial k nearest neighbor retrieval may benefit by using a feature subset optimized for the entire data set. However, on subsequent retrievals, we want to provide the decision tree inducer with all of the features. The inducer is thus not limited to selecting features for splitting from the subset and can create the tree best customized to the feedback data. Indeed, a subset chosen for the entire dataset will be biased toward retrieval of classes with greater instance membership and may perform poorly for queries belonging to minority classes.

3.5. The RFDT + Algorithm

A variation of our original algorithm automatically searches for feature subsets as an online step to improve retrieval performance from databases whose instances have class labels. The algorithm is identical to the RFDT algorithm when used in the hybridized mode with Subset A up to the point where the instances in relevant nodes are pooled and ready for retrieval. Here, instead of using a nearest neighbor retriever to retrieve from these instances, SFS is run on this pool to select the feature subset with maximal precision. Next, a nearest neighbor retrieval is done on the pool, using this subset. Thus, this feature selection attempts to discover a feature subset that best discriminates the classes with respect to the pooled relevant instances.

4. EMPIRICAL RESULTS

The performance of the relevance feedback decision tree retrieval algorithm was benchmarked against that of another recent image retrieval system known as the probabilistic feature relevance learner, developed by Peng *et al.* [10]. Details of the image database, experimental setup, and experimental results are provided in this section.

4.1. The HRCT Image Database

The image database upon which our experiments are based is populated by 1004 HRCT gray-scale images of human lungs at a resolution of 256×256 pixels. An example image is shown in Fig. 2. Each image contains at least one form of lung disease such as centrilobular emphysema or bronchiectasis. The database contains images that are diagnosed by a lung specialist as one of 22 disease classes. (Note that this disease class information is not used in the retrieval process.) Each image contains one or more subregions delineated by a radiologist, each one labeled with its disease class. These regions are called pathology-bearing regions, or PBRs. Image feature extraction is performed local to each PBR as well as on the entire lung. These two vectors are concatenated to form the full feature

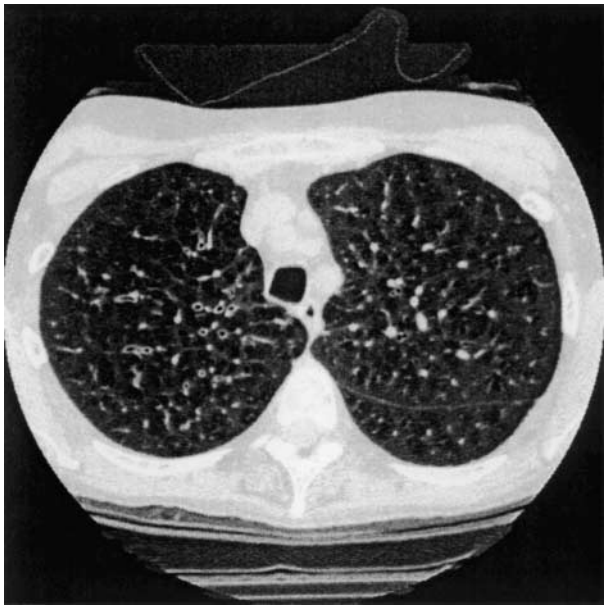

FIG. 2. An HRCT scan of a human lung.

vector for the PBR [17]. Of the 1916 PBRs, 1406 of them belong to one of four disease classes: MCTD, bronchiectasis, hemorrhage, and centrilobular emphysema. (Centrilobular emphysema, occurring in 654 PBRs, is the majority class.) The remaining 18 disease classes occur in less than 100 PBRs each.

Our feature extraction software extracts 184 features, as given in Table 1. These include domain independent properties such as gray-scale histogram and texture, and domain specific properties that relate to PBR geometry, among others [18]. Because these features have different scales, all feature vectors are normalized to have zero mean and unit standard deviation before being entered into the database. This normalization is crucial in preventing certain features from having an initial bias toward being more relevant than others when being compared with a distance metric. The retrieval algorithms operate on feature vectors, not, technically, on images. Thus, when we say that an algorithm retrieves an image, we really mean that it retrieves the vector associated with that image and then presents the user with the image.

4.2. Experimental Design

Our experiments compare our feedback decision trees retriever to the probabilistic feature relevance learning (PFRL) using identical operating conditions. PFRL is a weighted K nearest neighbor retriever that adjusts feature weights based on user feedback and has been shown to be one of the better retrievers in existence. PFRL is controlled by two parameters: C, a bias–variance tradeoff, and T, the learning rate. The C parameter may take integer values between 1 and $K-1$, while the T parameter may be set to any positive real number. Experimentation illustrated that a value of $T=4$ optimized the retriever's precision with respect to our database. The precision is insensitive to the choice of C; it was set to ceiling$(K/2)$ with good results. Our relevance feedback decision tree retriever does not have parameters, and, therefore, no such optimization steps are necessary.

Our experimentation is divided into two parts. In the first section, there is no actual human relevance feedback provided to the system because we simulate feedback information, and the second section contains data from a trial involving a radiologist. The motivation for offline experiments is straightforward. User relevance feedback must be made by a human expert, a radiologist in this domain, but this feedback is expensive in both time and effort. A person unfamiliar with lung HRCT imagery will not provide useful feedback to the system and furthermore will be of no use in evaluating a system's performance. We do, however, exploit the fact that we have a disease class label for each image. We make one reasonable assumption about user preferences for the sake of system evaluation: if the disease class of the query is A, then a typical user would be most interested in seeing images that are also diagnosed as having disease A. Thus, for the offline experiments, if an image matches the disease class of the query, it is considered relevant; otherwise, it is considered irrelevant.

The ability of the system to return duplicate images during feedback iterations for a given query is a parameter in our experimentation. Initial simulated user results retrieved images with replacement: one image could surface multiple times during the retrievals for a query. This practice is common with systems developed elsewhere [10] which led us to follow suit to enable a consistent comparison of our system to these others. Unfortunately, allowance of these repeated retrievals has the potential to overinflate precision measurements. Furthermore, in the human user scenario, it wastes the user's time because he or she needs to view and provide feedback on the same images multiple times. This repetition was prevented in all but the earliest simulated user experiments and was prevented in the human user experiments.

Our database contains 1916 feature vectors extracted from the image set, each corresponding to a PBR. (There are, on average, one to two PBR's per image.) A group of images often may be from the same patient. (HRCT scans of different cross-sections of the lung are called *slices*.) It is important to note that in the evaluation trial, our retrieval system returns images that have some similarity to the query but are not from the same patient as the query. All experiments prevent retrieval of images whose source patient matches the query patient.

In our experiments we present results for $K = 4$ and $K = 10$. We argue that fewer than four marked images per retrieval will degrade the performance due to insufficient feedback and that more than 10 iterations would be unrealistic in terms of the satisfaction of the user. We call the initial retrieval an *iteration*, even though in strict language an iteration is a repetition. While substantial research [10, 13] has demonstrated the efficacy of systems that routinely use $K > 10$, we feel that marking such a large number of images per retrieval is an excessive burden for the user. We evaluate the systems over 10 iterations, which is an upper bound on a plausible number of iterations expected in practice.

4.2.1. Simulated user experiments. In these experiments, each retriever was run 1916 times, once for each PBR in the database. For a given query, we recorded the retrieval precision, which we computed to be the fraction of retrievals whose class matches that of the query. The simulated experiments are divided into two subgroups: those in which we allow repeated retrieved images and those in which we do not. The work allowing repeated retrieved images predates the work allowing different feature subset configurations; it is run using Subset A for both initial and subsequent retrievals. The experiments preventing repeated retrievals is given a more thorough treatment. It was operated in the three modes

outlined in the previous chapter: using all features all the time, using a feature subset all the time, and using a subset for the initial retrieval but all features thereafter. We present experiments with both Subset A and B in the second two modes.

4.2.2. Human user experiments. Because our domain is comprised of HRCT images of the lung, the retrieval quality can only be assessed accurately by an expert in this domain: a physician who specializes in thoracic radiology. Such a specialist performed clinical trials of our system remotely via ASSERT, a Web-based interface for browsing and retrieval from our image data [19].

Given a query, the radiologist provided four rounds of iterative relevance feedback. She could elect to rate each retrieved image as either "Strongly Agree," "Agree," "Not Sure," "Disagree," or "Strongly Disagree." Note that the current incarnations of the retrievers compared only handle binary relevance feedback; thus, "Strongly Agree" and "Agree" were assigned to "relevant" and the remaining selections were assigned to "irrelevant" during communication of the ratings to the retrievers.

The experiment was divided into two phases. In the first phase, 19 images were selected at random from the database to be used as queries. For each query, either RFDT or PFRL was selected as the retriever pseudo-randomly; the selection keyed off of a character in the image filename. The retriever type was hidden at all times from the radiologist, thus making the experiment single-blind. There was no need to "pad" extra time into the (faster) PFRL retriever because retrieval time overhead associated with the Web interface was great enough to make the retrieval times indistinguishable by a human. In the second phase, the same 19 queries were presented, but the keying was complemented—each query that used RFDT in the first phase now used PFRL and vice versa, providing feedback data for each and every query by both retrievers. During both phases, K was set to four. The radiologist did not have enough time to evaluate 19 queries with $K = 10$. For each round of relevance feedback, the frequency with which the radiologist provided each of the five ratings was logged.

4.3. Experimental Results

The results of the experiments where repeated retrievals are allowed are displayed in Fig. 3 for $K = 4$ and $K = 10$. Note that, for both values of K, the retrieval precision on the

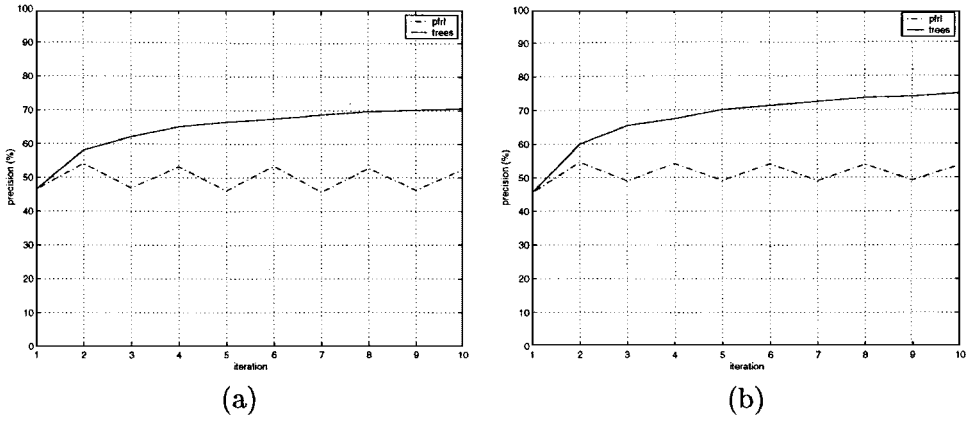

FIG. 3. Mean retrieval precision with repeated retrievals allowed for (a) $K = 4$ and (b) $K = 10$.

first iteration is identical for PFRL and RFDT because both default to a simple unweighted K nearest neighbor retrieval.

The oscillatory behavior of the mean precision of the PFRL retriever with respect to iteration indicates that the weights that it is storing are oscillating as well. This could have been dampened somewhat by decreasing the learning rate T, but we chose T to be the value that produced the best mean precision results on this domain. The accumulation of knowledge by the RFDT retriever is evidenced by the asymptotic behavior of its precision curve. Conversely, the PFRL system only uses current knowledge—the current set of retrieved images—to alter its next retrieval technique.

In the remainder of the experiments we do not permit repeated retrievals per query. Here we introduce a statistic more useful than precision for evaluating image retrieval: the cumulative sum of unique relevant images viewed up to and including iteration i. (Note that, in the no repetition case, this is a straightforward summing of the first through ith precisions with a scale factor K.) If we wanted to calculate cumulative recall, we need only divide these cumulative sums by the total number of relevant images in the database. Finding the total number of relevant images in a large database is, however, impractical. Imagine a person trying to retrieve 20 images of birds from a large database, like the Web. A Web feedback search engine may index thousands of bird images which would be marked relevant, but finding all of the relevant images through iteration would take hours per query. Even if calculating recall was less expensive, it would not have utility in our evaluation. We measure systems in their ability to retrieve the greatest number of relevant images in the fewest number of iterations.

Table 4 provides a summary of performance for $K = 4$. Note that the RFDT retriever is able to retrieve more relevant images after 10 iterations in all modes of retrieval operation. Note that the initial retrieval's mean relevant image count (the initial precision, scaled) is highest when using all the features. This is counterintuitive, because Subset A was designed to increase this number. A closer examination provides some insight. Our feature subset selection was wrapped around a 1 nearest neighbor retriever, but here $K = 4$. Due to the design of the subset selection criterion, in a 1 nearest neighbor case ($K = 1$), performance would necessarily increase, or at least stay the same, as compared to a 1 nearest neighbor retrieval experiment using all of the features. Unfortunately, no guarantees are made about other values of K—we are applying one criterion for subset selection and using the system in a way inconsistent with that criterion.

Subset B was designed to not necessarily increase initial precision, but to hopefully, in the end, allow more relevant images to be discovered by giving the retriever the most cases where

TABLE 4
Cumulative Retrieval Results with $K = 4$

		After 10 iterations	
Retrieval mode	After 1 iteration	PFRL	RFDT
All features	2.2	17.9	21.3
Subset A features	1.9	15.0	20.6
A/All hybrid	1.9	16.6	21.3
Subset B	1.7	11.1	18.6
B/All hybrid	1.7	15.4	21.2

TABLE 5
Cumulative Retrieval Results with $K = 10$

Retrieval mode	After 1 iteration	After 10 iterations	
		PFRL	RFDT
All features	5.1	46.2	50.5
Subset A features	4.5	38.5	49.4
A/All hybrid	4.5	43.9	50.5
Subset B	4.1	30.2	42.9
B/All hybrid	4.1	43.3	50.4

at least one image in the retrieved set is relevant. There is simply little evidence that, given this feature set and this image domain, the at-least-one subset selection provides a benefit.

Both retrievers do their respective best when using all of the features which indicates that, at least in this framework, feature subset selection integrated into relevance feedback CBIR techniques does not improve performance. Table 5 shows similar results for $K = 10$. There is no evidence that using $K = 4$ is causing learning deficiencies for either retriever besides the obvious fact that with $K = 10$, more potentially relevant images can be viewed in as many iterations.

Figure 4a shows the precision per iteration in the $K = 4$ case. The precision curves do not arc upward nearly as steeply because repeated retrievals are not allowed. We can now see that the RFDT retriever consistently averages over two relevant images per retrieval (50% precision) across the iterations while the PFRL retriever is unable to keep finding relevant images at that rate. Immediately after the first feedback session, however, the PFRL retriever does excel, as given by the spike at iteration number two. Figure 4b shows the cumulative relevant images found at a given iteration. Graphs for $K = 10$ show similar trends [8].

The RFDT+ algorithm was used to retrieve from the data in offline trials with $K = 10$. Recall that RFDT+ uses Subset A for the initial retrieval. We therefore compare its performance

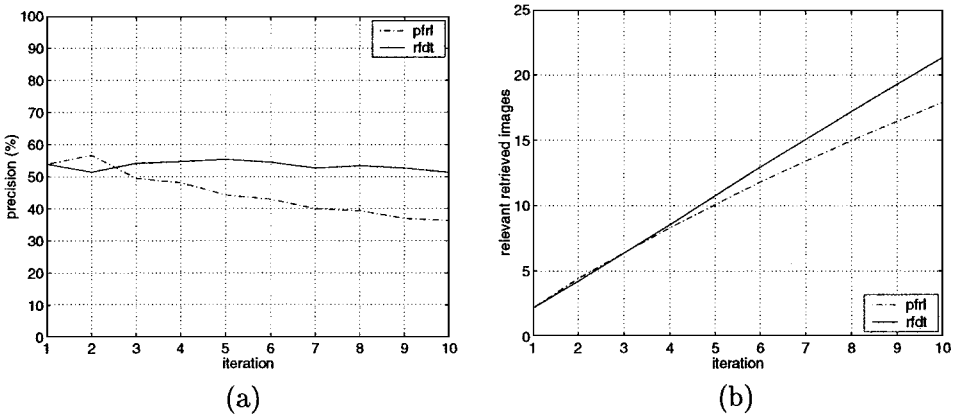

FIG. 4. Using all features, performance of PFRL, RFDT for $K = 4$. (a) Mean precision per iteration; (b) cumulative relevant retrievals.

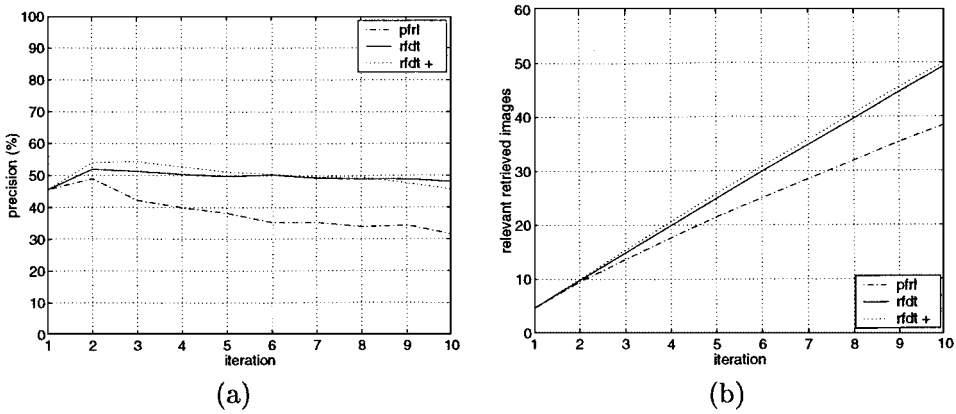

FIG. 5. Using all features, performance of PFRL, RFDT, RFDT+ for $K = 10$. (a) Mean precision per iteration; (b) cumulative relevant retrievals.

to the RFDT and PFRL retrievers in the cases where they use Subset A for the initial retrieval. Figure 5 shows the per iteration and the cumulative precision. Note that there is some benefit to using a feature subset tailored to a particular relevant instance pool for retrieval from that pool. More relevant images are uncovered in the first three retrievals than any other retriever, but the regular RFDT retriever performs slightly better retrieval during later iterations. The RFDT+ retriever discovers more relevant instances earlier, so fewer exist later.

Because the disease class distribution of our database is skewed, retrieval precision with respect to each class is not uniform. We report the retrieval results by class in Table 6 for $K = 4$. In 10 of the classes which constitute approximately 10% of the database's images,

TABLE 6
Per Class Retrieval Precision: Feature Subset A Modes with $K = 4$

		Full set		Subset A		A/Full hybrid	
Class	Frequency	PFRL base %	RFDT Δ%	PFRL Δ%	RFDT Δ%	PFRL Δ%	RFDT Δ%
1	654	82.5	+4.0	−14.3	−2.0	−3.0	+3.0
2	383	35.5	+15.8	−4.0	+19.3	−4.3	+16.3
3	234	43.3	+13.8	+0.5	+18.5	−2.3	+14.3
4	135	20.8	+13.5	−7.0	+16.0	−3.5	+13.8
5	74	1.3	+9.5	0.0	+6.0	0.0	+9.3
6	64	11.3	+12.8	−5.0	+9.3	+0.8	+13.0
7	61	0.5	+3.5	−0.5	+6.0	−0.5	+3.0
8	58	38.5	−8.3	−2.0	−4.5	−1.0	−7.5
9	57	27.3	+13.8	−27.3	−17.5	−27.3	+12.0
10	37	10.5	+7.3	−3.3	+9.8	−2.0	+8.0
11	34	6.3	+13.0	−5.0	+2.0	−3.5	+11.3
12	27	0.0	0.0	0.0	0.0	0.0	0.0
13	17	0.0	0.0	0.0	0.0	0.0	0.0
14	15	0.0	+0.3	0.0	0.0	0.0	+0.3
15–22	66	0.0	0.0	0.0	0.0	0.0	0.0

neither of the two retrievers was ever able to return even one relevant (same class as query) image. We lack the necessary features at this time to differentiate these minority classes from the majority classes. This result may be partially dependent on our SFS feature selection method, which attempts to select features with respect to overall classification accuracy, not mean class accuracy. For this reason, SFS tends to be biased toward selecting features that best distinguish the majority class from other classes. With $K = 10$, the base K nearest neighbor mean retrieval precision for the majority class is 73.7%. (Results for the $K = 10$ case closely mimic the results for the $K = 4$ case; a table for $K = 10$ is omitted for the sake of brevity.) PFRL has a mean precision of 82.4% on this class after 10 iterations and RFDT has a mean precision of 97.3% on this class after 10 iterations. Note that 100% retrieval is often difficult to attain and in some cases is theoretically impossible. For instance, if K is greater than the total population of images that are from the same class as the query, then precision cannot be perfect. This situation arises with skewed class distributions in which minority classes have few instances.

Notice that the RFDT retriever outperforms the PFRL for all but one class when using all of the features. (It is speculative to conclude that PFRL would surely outperform RFDT on this class with a larger database that has equal class distributions, thus rectifying any data sparseness effect that may be intervening here.) From this, we conclude that RFDT is not earning its precision through majority class retrieval while ignoring minority classes. In the next pair of columns, the majority class gets most hurt in absolute numbers by requiring the retrievers to use Subset A. PFRL relevancy percentages all decrease except for one class, while RFDT has a balance of losses and gains. This is also true for performances with the hybrid case, where the retrievers use Subset A for the first retrieval and all features for subsequent retrievals.

Figure 6 plots the results for the human user experiments. These cumulative statistics are averages across all 19 queries. Higher values on plots a and b and lower values on plots d and e characterize a superior retriever. RFDT retrieves more "Strongly Agree" rated images as well as fewer images in the "Disagree" and "Strongly Disagree" categories, but it does not retrieve as many images as PFRL that are rated "Agree." Recall that the retrievers do not use feedback that is this finely graded. To each of the retrievers, either of the first two ratings are considered relevant, and any of the latter three are considered irrelevant. If we lump the statistics into these coarser numbers, we find that the RFDT retriever accumulated a mean of 6.84 relevant images and 9.16 irrelevant images after all iterations. The PFRL retriever retrieves 6.52 relevant images and 9.48 irrelevant images. Thus, the RFDT retriever outperforms PFRL when the ratings are thresholded in accordance with how the retrievers threshold the feedback information themselves.

The results are less dramatic for this trial than for the offline experiments. This is attributable to the problem that for some images neither retriever is able to improve upon the initial incorrect results (this happens because several of our classes are sufficiently underrepresented that the features used for the initial retrieval are not relevant to these classes, as was discussed in Section 4.3).

A point which applies to all experimentation is that both retrievers are sufficiently fast for online implementation. On a Sun Ultra 1, the RFDT retriever takes approximately 1–2 s to perform a retrieval on average while the PFRL retriever takes approximately 300 ms. It should be noted that the speed of the RFDT retriever could be increased by using a data structure that allows for faster access than the simple linear search employed here.

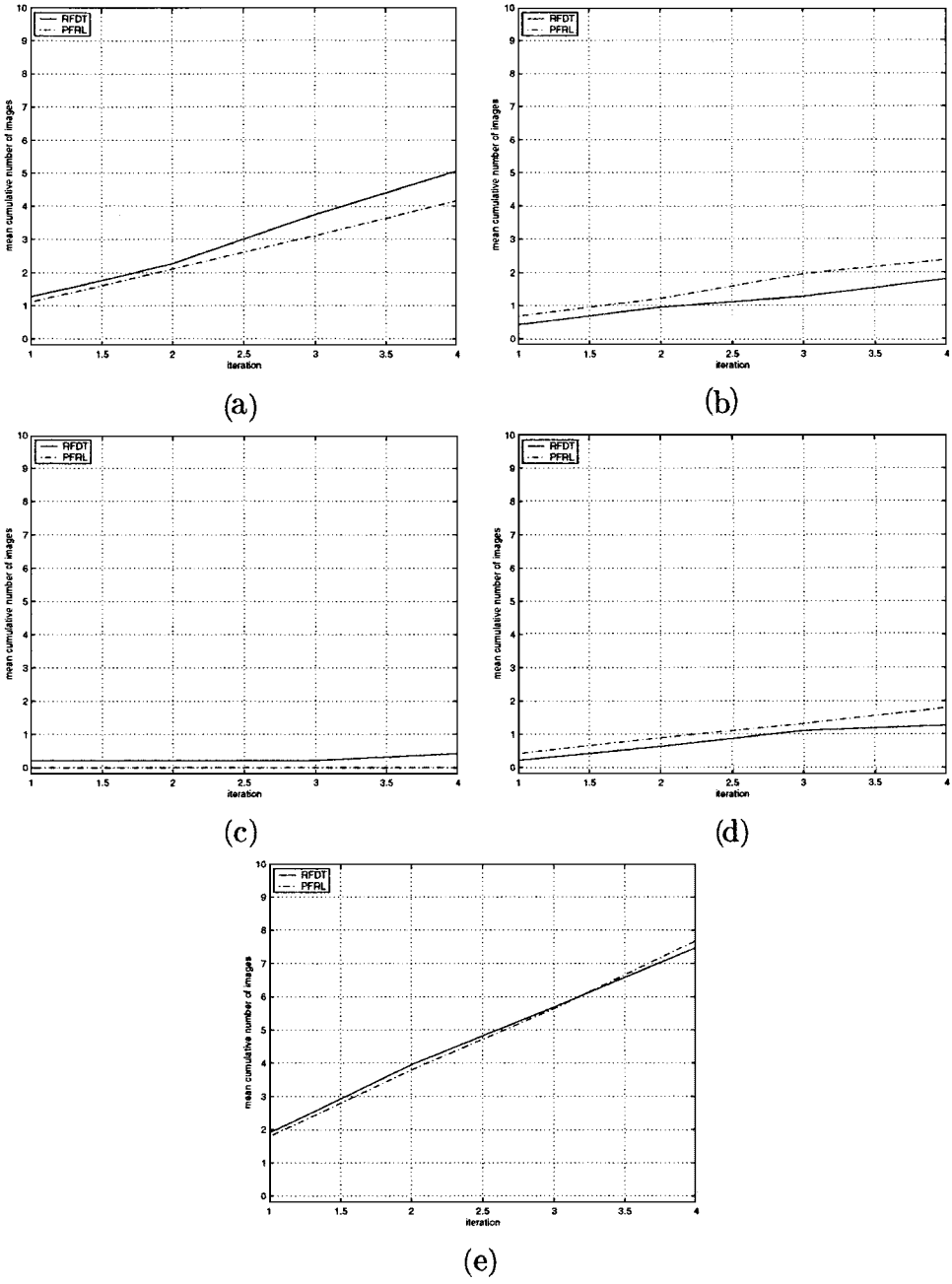

FIG. 6. Images rated (a) "Strongly Agree," (b) "Agree," (c) "Not Sure," (d) "Disagree," and (e) "Strongly Disagree."

5. CONCLUSION AND DISCUSSION

In this paper, we have presented a retrieval system called RFDT that improves its performance by inducing decision trees from user relevance feedback information. The decision trees are used to classify database instances into either class "relevant" or "irrelevant," and

a nearest neighbor retriever is used to return a ranked retrieval from the relevant pool to the user. This work has shown that using feature subsets for a query's initial or later retrievals provides, at best, modest performance gains on our medical image database. However, use of the RFDT+ algorithm, which performs an online sequential forward features selection just prior to retrieval from the relevant pool, provides small precision gains on the same data. In addition, the system retrieves any given image from the database at most once during a retrieval session. This practice, which is common in the more recent literature, prevents any ambiguity in evaluating retrieval precision across iterations.

Our system maintains a log of all of the feedback information and uses all of it in its model induction for a given retrieval iteration. Our system's use of this explicit data has proven more effective in practice than the implicit encoding that exists in feature weighting schemes. We have demonstrated that this long-term memory improves performance significantly with respect to a competing system that has no such faculty. Our system is novel in that it can take advantage of a database in which the database instances have class labels; the RFDT+ version of the retriever performs online SFS to enhance performance in this scenario. In all operating modes, our system remains competitively efficient with respect to space and computational time constraints.

There exist a number of open research issues not only in content-based image retrieval, but in information retrieval in general. As an example, retrieval in skewed class distributions is difficult because machine learning techniques often have an inductive bias that favors retrieval of the majority class. We have examined the systems' performances with respect to class, but that was purely to confirm that this bias existed for our data. A system that can adapt to the adverse environment created by a skewed class distribution would be a major breakthrough. Another issue concerns feature space exploration. Many retrieval systems take the conservative approach of exhausting all images in a locale of the feature space before looking elsewhere, but this may lead to suboptimal results. A gold mine of relevant instances could a short distance away in the feature space, but the overly conservative retriever would find them much later or not at all. The random retriever would not suffer from the same ill as the conservative retriever, but the random retriever is, clearly, far from optimal. Successful retrievers of tomorrow will need to develop policies that balance searching in tried and true areas with exploration beyond the frontier of the familiar.

ACKNOWLEDGMENT

This work was supported by the National Science Foundation under Grant IRI9711535 and the National Institutes of Health under Grant 1 R01 LM06543-01A1. We thank Chi-Ren Shyu, Jennifer Dy, and Christina Pavlopoulou for their generous donation of time and expertise.

REFERENCES

1. Y. Chen, X. Zhou, and T. Huang, One-class SVM for learning in image retrieval, in *Proceedings of the IEEE International Conference on Image Processing, 2001.*
2. I. J. Cox, M. L. Miller, T. P. Minka, and P. N. Yianilos, An optimized interaction strategy for Bayesian relevance feedback, in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition, Santa Barbara, California, 1998*, pp. 553–558.
3. J. G. Dy, C. E. Brodley, A. Kak, C. Shyu, and L. S. Broderick, The customized-queries approach to CBIR, in *SPIE Storage and Retrieval for Image and Video Databases VII*, Vol. 3656, pp. 22–32, 1999.

4. Y. Ishikawa, R. Subramanya, and C. Faloutsos, *Mindreader: Querying Databases through Multiple Examples*, Carnegie Mellon University Technical Report CMU-CS-98-119, 1998.
5. J. Kittler, Feature selection and extraction, in *Handbook of Pattern Recognition and Image Processing*, Academic Press, New York, 1986.
6. W.-C. Low and T.-S. Chua, Color-based relevance feedback for image retrieval, in *Proceedings of the International Workshop on Multimedia DMBS, 1998.*
7. S. D. MacArthur, C. E. Brodley, and C. Shyu, Relevance feedback decision trees in content-based image retrieval, in *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries, 2000.*
8. S. D. MacArthur, *Relevance Feedback Decision Trees in Content-Based Image Retrieval*, Masters thesis, School of Electrical and Computer Engineering, Purdue University, 2000.
9. T. P. Minka and R. W. Picard, *Interactive Learning Using a Society of Models*, MIT Technical Report 349, 1995.
10. J. Peng, B. Bhanu, and S. Qing, Probabilistic feature relevance learning for content-based image retrieval, *Comput. Vision Image Understand.* **75**, 1999, 150–164.
11. J. R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.
12. Y. Rui, T. S. Huang, and S. Mehrotra, Content-based image retrieval with relevance feedback in MARS, in *Proceedings of the IEEE International Conference on Image Processing, 1997.*
13. Y. Rui, T. Huang, M. Ortega, and S. Mehrotra, Relevance feedback: A power tool for interactive content-based image retrieval, *IEEE Transactions on Circuits and Video Technology, 1998.*
14. Y. Rui and T. Huang, Optimizing learning in image retrieval, in *Proceedings of Computer Vision and Pattern Recognition*, IEEE Computer Society Press, Los Alamitos, CA, 2000.
15. T. Seidl and H. Kriegel, Efficient user-adaptable similarity search in large multimedia databases, in *Proceedings of Very Large Databases, 1997.*
16. C. E. Shannon, A mathematical theory of communication, *Bell System Tech. J.* **27**, 1948, 379–423.
17. C. Shyu, C. E. Brodley, A. C. Kak, A. Kosaka, A. Aisen, and L. Broderick, Local versus global features for content-based image retrieval, in *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries, 1998.*
18. C. Shyu, C. E. Brodley, A. Kak, A. Kosaka, A. Aisen, and L. Broderick, ASSERT, A physician-in-the-loop content-based image retrieval system for HRCT image databases, *Comput. Vision Image Understand.* **74**, 1999, 111–132.
19. C. Shyu, A. C. Kak, C. E. Brodley, and L. S. Broderick, Testing for human perceptual categories in a physician-in-the-loop CBIR system for medical imagery, in *Proceedings of the IEEE Workshop of Content-Based Access of Image and Video Databases, 1999.*
20. L. Taycher, M. La Cascia, and S. Sclaroff, *Image Digestion and Relevance Feedback in the ImageRover WWW Search Engine*, Boston University Technical Report BU CS TR97-014, 1997.
21. K. Tieu and P. Viola, Boosting image retrieval, in *Proceedings of Computer Vision and Pattern Recognition*, IEEE Computer Society Press, Los Alamitos, CA, 2000.
22. S. Tong and E. Chang, Support vector machine active learning for image retrieval, in *ACM Multimedia, Ottawa, Canada, 2001.*
23. C. J. van Rijsbergen, *Information Retrieval*, Butterworths, London, 1979.
24. X. S. Zhou and T. S. Huang, Exploring the nature and variants of relevance feedback, in *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries, in conjunction with CVPR, 2001.*
25. X. S. Zhou and T. S. Huang, Small sample learning during multimedia retrieval using BiasMap, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2001.*