# Speeding Up Active Relevance Feedback with Approximate $k$NN Retrieval for Hyperplane Queries

Michel Crucianu* Daniel Estevez† Vincent Oria‡ Jean-Philippe Tarel§

March 27, 2008

## Abstract

In content-based image retrieval, relevance feedback is a prominent method for reducing the "semantic gap" between the low-level features describing the content and the usually higher-level meaning of user's target. Recent relevance feedback methods are able to identify complex target classes after relatively few feedback iterations. However, since the computational complexity of such methods is linear in the size of the database, retrieval can be quite slow on very large databases. To address this scalability issue for active learning-based relevance feedback, we put forward a method that consists in the construction of an index in the feature space associated to a kernel function and in performing approximate $k$NN *hyperplane* queries with this feature space index. The experimental evaluation performed on two image databases show that a significant speedup can be achieved, at the expense of a limited increase in the number of feedback rounds.

*Keywords*: content-based image retrieval, relevance feedback, scalability, M-tree, approximate search, hyperplane query.

## 1 Introduction

The query-by-example paradigm for content-based retrieval suffers from the discrepancy—known as "semantic gap"—between the low-level features that can be readily extracted from documents and higher-level descriptions that are meaningful for the users. By including the user in the retrieval loop, relevance feedback (RF) is in many cases able to bridge this semantic gap. First introduced for the retrieval of text documents (Rijsbergen, 1979), RF rapidly developed for image retrieval ((Chang et al., 2003), (Crucianu et al., 2004)), mainly because a user can be very fast in evaluating the relevance of an image.

An RF session is divided into several consecutive rounds (or iterations); at every round the user provides feedback regarding the retrieval results, e.g. by qualifying items[1] returned as either "relevant" or "irrelevant"; from this feedback, the search engine learns the features describing the relevant items and returns improved results to the user. An RF mechanism employs a *learner* and a *selector*. At every feedback round, the user marks (part of) the items returned by the search engine as "relevant" or "irrelevant". The learner exploits this information to reestimate the target of the user. Since the number of examples is typically very low, this stage is very fast with such learners as support vector machines (SVM, (Schölkopf and Smola, 2002)) and cannot be considered a scalability challenge. With the current estimation of the target, the selector chooses other items for which the user must provide feedback during the next round.

Several recent RF methods rely on *active learning* (Tong and Koller, 2000), (Tong and Chang, 2001): the user is asked to provide feedback on those items that can maximize the transfer of information from the user to the system. This typically implies that the selector must return, at every feedback round, the *most ambiguous* items (if possible, with a complementary condition of low redundancy, as in (Ferecatu et al., 2004)), i.e. those items that are closest

*Vertigo-CEDRIC, CNAM, 292 rue St. Martin, 75141 Paris Cedex 03, France, Michel.Crucianu@cnam.fr

†Vertigo-CEDRIC, CNAM, 292 rue St. Martin, 75141 Paris Cedex 03, France, Esteve_D@auditeur.cnam.fr

‡Department of Computer Science, New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA, Vincent.Oria@njit.edu

§Laboratoire Central des Ponts et Chaussées, 58 Bd. Lefebvre, 75015 Paris, France, Tarel@lcpc.fr

[1] While our evaluations here only concern image databases, RF in general and our method in particular are not specific to such databases, so we prefer to speak about "items" rather than images.

to the frontier of what the system considers (at the current round) to be the target class. If the selector has to evaluate the ambiguousness of all the items in the database then this approach does not scale to very large databases. Scalability requires an index structure able to make the complexity of the selection stage sublinear in the size of the database. Existing multidimensional indexes and associated search methods (Samet, 2006) mostly concern similarity-based retrieval with queries having the same nature as the items stored in the database. Since a decision frontier is different in nature from the items in the database, such proposals cannot be directly applied; new solutions must be developed for answering *frontier-based queries*. The decision frontier can be complex in the space where the items are described, making the retrieval problem difficult to formulate in this initial space. However, by the use of specific kernel functions (Berg et al., 1984) the initial space is mapped to a corresponding *feature space*, where a hyperplane can be an appropriate decision frontier (Schölkopf and Smola, 2002).

Our main contribution here is to put forward a retrieval method that consists in performing approximate $k$NN *hyperplane* queries with an index built in this feature space. Since the index and the query processing algorithms are defined in the feature space, they do not depend on the input space. The method can thus be directly applied to various types of data as long as an appropriate kernel exists for the corresponding input space. Also, this method is not limited to the RF context and can be employed for active learning in general. The evaluation performed on two image databases, one of which contains 110,250 images, shows that a significant speedup can be achieved, at the expense of a limited loss in retrieval precision. These results also point out that approximate search methods can behave well in very challenging conditions: high-dimensional spaces, with queries expected to have a low selectivity.

The next section summarizes the background of this work; after describing the SVM-based active RF method we attempt to accelerate, we regard existing proposals for the scalability of recent RF methods; the section ends with a brief presentation of approximate $k$NN retrieval with point queries in an M-tree index, which is the starting point of the scalability solution put forward here. The feature space M-tree and the corresponding $k$NN retrieval algorithms for hyperplane queries are then defined in Section 3. An experimental evaluation of the proposed method on two ground truth image databases is provided in Section 4. Finally, several issues regarding approximate retrieval and the properties of various kernel functions are discussed in Section 5.

# 2 Background

## 2.1 Relevance feedback with support vector machines

Part of the recent work on RF (e.g. (Tong and Chang, 2001), (Peng and Heisterkamp, 2003), (Ferecatu et al., 2004), (Tao et al., 2006), (Tao et al., 2008)) is based on support vector machines (SVM) because they avoid too restrictive assumptions regarding the data, are very flexible and allow fast learning with a reasonably low number of examples.

Support vector machines belong to the family of kernel methods (Schölkopf and Smola, 2002), who first map the data from the original (input) space $\mathcal{I}$ to a higher-dimensional *feature space* $\mathcal{H}$ and then perform linear algorithms in $\mathcal{H}$. The nonlinear mapping $\phi : \mathcal{I} \to \mathcal{H}$ is implicitly defined by a kernel function $K : \mathcal{I} \times \mathcal{I} \to \mathbb{R}$ endowing $\mathcal{H}$ with a Hilbert space structure if the kernel is positive definite (Berg et al., 1984). The inner product $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ can be expressed as $\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle = K(\mathbf{x}, \mathbf{y})$. This "kernel trick" allows to reduce inner products in $\mathcal{H}$ to ordinary kernel computations in $\mathcal{I}$ and thus extend linear algorithms relying on inner products in $\mathcal{H}$ to nonlinear algorithms based on more ordinary computations in $\mathcal{I}$. For all the experiments presented in Section 4, $\mathcal{I} = \mathbb{R}^d$, but the method put forward here is not restricted to this case. The input space $\mathcal{I}$ does not even need to be a vector space, as long as a positive definite kernel can be defined (Berg et al., 1984).

The class of kernels for which SVM algorithms hold can actually be enlarged to the so-called *conditionally* positive definite (cpd) kernels (Schölkopf, 2000). While the main line of presentation of our retrieval method is restricted to positive definite kernels, the extension to a large family of cpd kernels is given in the appendix.

One-class SVM were put forward as a means to describe the domain of a data distribution having a potentially complex description in an input space $\mathcal{I}$. The data is first mapped to the feature space $\mathcal{H}$. Then, in the first formulation, given in (Tax and Duin, 1999), the smallest sphere in $\mathcal{H}$ that contains the images of the data items is taken as the feature space representation of the domain of the distribution

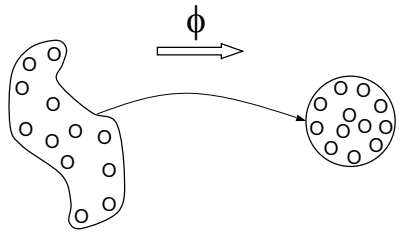


Figure 1: Domain description with 1-class SVM. $\phi$ maps input space (left) to feature space (right).

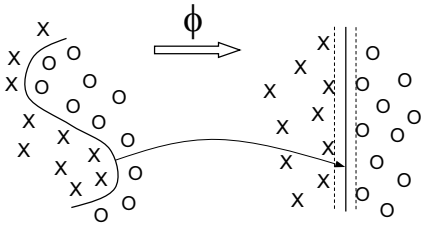


Figure 2: Discrimination with 2-class SVM. $\phi$ maps input space (left) to feature space (right).

(Fig. 1). This sphere is defined by a center and a radius; the center is a linear combination of the images of (part of) the data items. Another formulation for 1-class SVM, where the domain of the distribution is defined by a hyperplane in feature space, was put forward in (Schölkopf et al., 2001). One-class SVM were used for RF in (Peng and Heisterkamp, 2003) to model the distribution of the positive examples (items marked as "relevant") and return the unmarked items whose images in feature space are the nearest to the center of the sphere (i.e. the items considered by the learner as potentially the most "relevant"). In this case, the information provided by the negative examples (items marked as "irrelevant") is ignored.

A *2-class* SVM aims to identify a frontier between two classes, based on a set of learning (labeled) examples. The 2-class SVM (Fig. 2) chooses as discrimination frontier the hyperplane in feature space that maximizes the *margin* to the examples from each of the 2 classes. This hyperplane is the feature space image of a usually nonlinear frontier in input space (depending on the kernel employed). The hyperplane is defined by an orthogonal vector and a position threshold. Since the orthogonal vector is in the subspace spanned by the $n$ vectors $\phi(\mathbf{x_i})$ ($\mathbf{x_i}$ being the original data points), it is expressed as a linear combination of the "support vectors", i.e. of those vectors who are within the margin. Learning consists in identifying the support vectors and computing the linear coefficients, which is done by a fast procedure for constrained quadratic optimization.

When used for RF, the 2-class SVM learns at every feedback round to discriminate the target class (of the "relevant" items) from the rest of the database. The SVM learner is trained using all the available examples, both positive (items marked as "relevant" by the user) and negative (items marked as "irrelevant"). Then, the selector must choose yet unmarked items for which the user should provide feedback during the next round.

The first advances in SVM-based relevance feedback focused on the selector. It was found in (Tong and Chang, 2001) that active learning could lead to a faster improvement in retrieval precision during consecutive iterations: the user can provide more information if presented with the most "ambiguous" items (the closest to the discrimination frontier) than with the most "positive" ones (the farthest from the frontier, on the "relevant" side). A further criterion aiming to minimize the redundancy between the selected items was added in (Ferecatu et al., 2004). In (Ferecatu et al., 2004) this joint criterion was named MAO, from "most ambiguous and orthogonal", since for positive definite kernels the items are non-redundant if their feature space representations are orthogonal. MAO was also successfully employed in (Ferecatu et al., 2008) for performing RF with combined visual and conceptual image features.

Significant improvements were then brought to the learner component of the RF mechanism. Several issues were addressed, including the small sample size problem that is severe for RF during the first iterations and the differentiated treatment of the "irrelevant" examples. Generalization can be very poor when only a small amount of high-dimensional labeled data is available; this difficulty can be alleviated by finding appropriate solutions for dimension reduction or for increasing the number of labeled examples. In (Tao et al., 2006), the random subspace method for feature selection was combined with ensemble learning (asymmetric bagging) to improve the stability of SVM learning in the early RF iterations and to balance "relevant" and "irrelevant" examples. Co-training, a semi-supervised learning method exploiting the complementarity of several learners for enhancing the training data, was jointly employed with the random subspace selection in (Li et al.,

2006). Note that the use of lower-dimensional features can also diminish the cost of kernel computations and the number of support vectors, thus contributing to a reduction in the cost of the selection stage (which nevertheless increases linearly with the size of the database). Motivated by the early analysis in (Zhou and Huang, 2001) highlighting the different nature of "relevant" and "irrelevant" feedback, a differentiated treatment of the items marked as "irrelevant" was proposed in (Tao et al., 2007) based on the kernel biased marginal convex machine. An alternative solution, introducing orthogonal complement component analysis, is suggested in (Tao et al., 2008). For both proposals, learning is performed in a reduced space, which brings in the corresponding advantages.

We address here the scalability of RF by attempting to make the complexity of the selection stage increase sublinearly with the size of the database. This relies on an index structure that is computed prior to any feedback rounds, so it cannot consider many alternative sets of features, as would be required by approaches that select different sets of features during the iterations of an RF session. This explains why we consider in the following the SVM-based active RF method using the MAO selection criterion.

To implement MAO, a larger set of ambiguous unmarked items is selected first. Then, the low redundancy (MAO) selection is built from this set by iteratively choosing as a new example the item represented by the vector $\mathbf{x}_j$ that minimizes the highest of the values taken by $K(\mathbf{x}_i, \mathbf{x}_j)$ for all the $\mathbf{x}_i$ examples already included in the current MAO selection: $\mathbf{x}_j = \text{argmin}_{\mathbf{x} \in S} \max_i K(\mathbf{x}, \mathbf{x}_i)$, where $S$ is the set of items not yet included in the current MAO selection and $\mathbf{x}_i$, $i = 1, \ldots, n$ are the already chosen candidates. Note that the items that are most likely to be "relevant" are those that are farthest from the current discrimination frontier, on the "relevant" side.

In the following we call "frontier" queries (FQ) those selecting the items whose feature space representations, $\phi(\mathbf{x})$, are closest to the hyperplane (on either side), and "maximum" queries (MQ) those selecting items whose feature space representations are farthest from the hyperplane, on the "relevant" side.

## 2.2 Index structures for relevance feedback

Before describing the new method, we shortly present here existing proposals addressing the scalability of RF. In (Peng and Heisterkamp, 2003) RF is performed with 1-class SVM. The support of the distribution of "relevant" examples is described in feature space by the smallest sphere containing the representations of these examples. The selector does not follow an active learning approach: it must simply return the unmarked items whose representations are the closest to this sphere. To do this, the authors suggest to build an M-tree in the feature space and to perform standard $k$NN retrieval using the center of the sphere as a query, which allows to speed up the selection stage by a factor of 2 to 3. Better results are obtained by the method proposed in (Heisterkamp and Peng, 2003), where a vector approximation file (VA-file) is defined in the feature space of a kernel (KVA-file) and employed with the same type of point-based queries. The VA-file method combines a sequential scan of the quantified data with a full search in the pre-selected data. Since the feature space induced by the kernel function can be infinite-dimensional, to build a VA-file the authors select a reduced set of orthogonal basis vectors in this space. Both solutions provide a speedup of the selection stage of every RF round. But since they do not exploit the information provided by the negative examples, 1-class SVM do not perform as well as 2-class SVM with active learning in identifying the target of the user.

A solution was proposed in (Panda and Chang, 2005), (Panda et al., 2006) for 2-class SVM with active learning. The selection stage for active learning appears to rely on the use of clustering in feature space and on the selection of the clusters that are nearest to the hyperplane corresponding to the discrimination frontier in order to answer FQs. A new index structure, KDX, is introduced for processing MQs: since for most of the kernels employed for RF one has $K(\mathbf{x}, \mathbf{x}) = \alpha$ for some fixed $\alpha$, the feature space representations of all the items in the database are on a hypersphere of radius $\alpha$. These representations are then uniformly distributed in rings around a central vector, and these rings are indexed according to the angle to the central vector. A second index is used within each ring. For a given MQ query, KDX performs intra and inter-ring pruning. KDX performs well for MQs, but the principle appears difficult to extend to FQs that are our main focus here.

Finally, in (Panda and Chang, 2006) the authors suggest to remain instead in the input space (space of the item descriptors) and to use an R-tree or a similar index structure to answer range queries correspond-

ing to the hyper-rectangles where either positive or negative examples were already found. However, the input space can also be high-dimensional (e.g., the dimension of typical image descriptors is between 20 and a few hundreds), so an R-tree can hardly be expected to perform well.

Section 3 presents the scalability solution we put forward for FQs, relying on a feature space M-tree and $k$NN retrieval algorithms for *hyperplane* queries. But first, we briefly remind in the following what an M-tree index is and how exact and approximate $k$NN retrieval is performed with such an index for point queries.

## 2.3 M-tree for exact and approximate search

The M-tree index structure was introduced in (Zezula et al., 1996), (Ciaccia et al., 1997) to address cases where the representation space is not a vector space, but is only endowed with a metric structure. The M-tree and the associated retrieval methods rely on the properties of a metric, the triangular inequality being especially important. We only provide here a very brief description of $k$NN retrieval, the reader should refer to (Zezula et al., 1996), (Ciaccia et al., 1997) for further details.

Let $Q$ be the query object (only point queries are handled), $O_p$ the parent object of the current node $N$, $O_r$ a rooting object belonging to node $N$, $r(O_r)$ the covering radius of $O_r$ and $T(O_r)$ the sub-tree having $O_r$ as root. Retrieval is based on the following results:

1. If $d(O_r, Q) > r(Q) + r(O_r)$, then it is safe to prune $T(O_r)$ since $d(O_j, Q) > r(Q)$ for each object $O_j$ in $T(O_r)$.
2. If $|d(O_p, Q) - d(O_r, O_p)| > r(Q) + r(O_r)$, then it is not even necessary to compute $d(O_r, Q)$ since $d(O_r, Q) \geq |d(O_p, Q) - d(O_r, O_p)|$, so $d(O_r, Q) > r(Q) + r(O_r)$.

To answer $k$NN (or "top-$k$") queries, the method in (Ciaccia et al., 1997), (Zezula et al., 1996) makes use of a priority queue, `PR`, of pointers to *active* sub-trees (where objects satisfying the query can potentially be found) and of a $k$-elements array, `NN`, for storing neighbors by increasing order of their distance to the query. Below, $d_{\min}(T(O_r)) = \max\{d(O_r, Q) - r(O_r), 0\}$ is the lower bound for the distance between any object in $T(O_r)$ and $Q$, $d_{\max}(T(O_r)) = d(O_r, Q) + r(O_r)$ is the upper bound for the distance between any object in $T(O_r)$ and $Q$, and $d_k$ is the largest distance in `NN`. The value of $d_k$ can be seen here as a *dynamic* search radius. Nodes in `PR` are sorted by increasing order of their $d_{\min}$ values.

The algorithm for answering $k$NN queries begins with an empty `NN` array, with `PR` containing the root node and with an infinite value for the dynamic search range $d_k$. When a node in the `PR` list is processed it is removed from `PR` and all its non-leaf children are added to `PR`. Objects (leaves in the M-tree) are progressively found and, if their distance to the query is lower than the current value of $d_k$, are introduced in the `NN` array (an object already in the array may have to be removed) and $d_k$ is updated. Search stops when the $d_{\min}$ of the first entry in `PR` (the entry with the lowest $d_{\min}$) is higher than the distance to the query of the $k$-th entry in `NN` (the entry having highest distance to the query): none of the remaining nodes in `PR` can contain a neighbor that is closer to the query than one of the $k$ neighbors already in `NN`. The use of the dynamic range $d_k$ and of the triangular inequality, supporting the second result in the previous enumeration, makes $k$NN retrieval with an M-tree more efficient than the method put forward in (Hjaltason and Samet, 1995).

To achieve yet faster retrieval with the M-tree, approximate search methods were developed. The Approximately Correct Nearest Neighbor (AC-NN) algorithm introduced in (Arya et al., 1998) was applied to the M-tree in (Ciaccia and Patella, 2000). By accepting as *approximate* NN for a query an object that is within a distance lower than $(1+\epsilon)d_k$ to the query, where $d_k$ is the distance to the true NN and $\epsilon > 0$, search can be stopped earlier, which produces a relative improvement in retrieval speed.

In (Ciaccia and Patella, 2000) the authors notice that $k$NN retrieval using the dynamic radius algorithm presented above can be described as having two stages: during the first stage, exact (or approximate) nearest neighbors are found; during the second stage, further nodes are retrieved from the priority queue `PR` in order to check whether the neighbors already found are the nearest indeed. Difficult retrieval cases are then considered, where the distribution of the distances between the items in the database is concentrated (but not to the point where NN queries become meaningless) and produces significant overlap in the M-tree, with the important consequence of reducing the selectivity in processing queries. In such difficult cases, the second stage can become much longer than the first because, given the significant node overlap, the priority queue still contains many candidate

nodes when the nearest neighbors are found; however, below a certain search radius, the probability of finding better neighbors in the remaining nodes becomes negligible. In brief, many candidate nodes intersect the search range but the intersections are empty.

The Probably Approximately Correct (PAC-NN) algorithm put forward in (Ciaccia and Patella, 2000) attempts to provide a solution to this problem by shortening the second stage of the retrieval process. Besides the use of the accuracy parameter $\epsilon$ defining the quality of approximation (as for AC-NN), a confidence parameter $\delta$ is introduced. The PAC-NN algorithm attempts to guarantee with probability at least $1-\delta$ that the "relative error" $\epsilon$ will not be exceeded by the approximate nearest neighbor returned. For this, search is stopped when the dynamic search radius becomes lower than the bound $(1+\epsilon)r_{q,\delta}$, where $r_{q,\delta}$ is such that $P(\exists o,\, d(q,o) \leq r_{q,\delta}) \leq \delta$ and has to be estimated from the data, prior to processing queries. PAC-NN generalizes both the AC-NN retrieval, obtained when $\delta = 0$, and the Correct Nearest Neighbor (C-NN) retrieval ($\epsilon = 0$ and $\delta = 0$). PAC-NN avoids searching "too close" to the query object and uses the distance distribution to the query object to derive a stopping criterion.

# 3 Hyperplane queries in an M-tree

## 3.1 Principle of the method

Relevance feedback is performed here using 2-class SVM and active learning. To speed up the selection stage of every feedback round, we adapt the M-tree to hyperplane queries in the feature space and make use of the kernel trick for reducing distance computations in the feature space to kernel computations in the input space. More specifically, an M-tree is built in the feature space (FSM-tree in the following) associated to the kernel and the $k$NN of the hyperplane that is the frontier given by the SVM (see Fig. 3) are retrieved. We speak of *hyperplane* queries (or, more generally, FQs).

Building the index structure in the feature space rather than in the input space has two potential benefits. First, the query has a simpler expression in the feature space: a hyperplane in the feature space usually corresponds to a complex nonlinear surface in input space. Second, the input space does not need to be a vector space: as long as a positive definite kernel can be defined (or specific conditionally positive definite kernels, see the appendix), a distance-based index like the M-tree can be used in feature space. Many such kernels were put forward for sets, sequences, graphs, etc. With all these kernels, index construction and index-based retrieval algorithms remain unchanged.

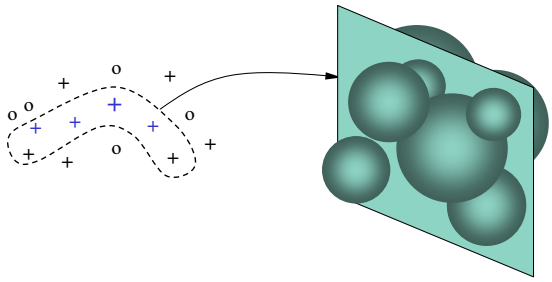

Figure 3: The frontier in input space (left) of the class of "relevant" images, as estimated by the SVM, is mapped to a hyperplane that is used as a query in the FSM-tree (right).

However, feature spaces are usually of much higher dimension than the input space. But since the dimension of the input space is already very high, the impact of this difference may not be very important. In fact, for several of the kernels employed in the following, the feature space is infinite dimensional; but the distribution of distances computed in the feature space is not very different from the distribution of distances in the input space, so metric indexing should be nearly as effective in feature space as in input space. It is important to note that, given the high dimension of both spaces, component-based indexing methods (SR-tree, VA-file, etc.) cannot be expected to perform well.

The use of hyperplanes as queries also raises several difficulties. First, dimension reduction methods cannot be employed: hyperplanes are not necessarily orthogonal to the reduced space, so items can be close to the query in original space and far from it in the reduced space; filtering in the reduced space can thus produce false negatives. The same difficulties occur if one attempts to use index structures that rely on various types of low-dimensional projections.

Second, a hyperplane query can be expected to be much less selective than a point. As will be seen later, approximate $k$NN retrieval does provide a practical solution to this problem. Third, for many index structures the computations involved can be complex;

the M-tree offers simple computations of the minimal or maximal distance between a node (defining a sub-tree) and a hyperplane

As an alternative, the $k$NN retrieval proposal in (Hjaltason and Samet, 1995), which also applies to queries having a spatial extension, could be considered and adapted for feature space indexing. Our choice of extending to hyperplane queries in a feature space the $k$NN retrieval with an M-tree (originally designed for point queries) was motivated by the comparatively higher efficiency of the $k$NN algorithm of the M-tree.

The first step of our method is the construction of an M-tree in the feature space (FSM-tree) associated to the kernel employed for the SVM. This requires the computation of Euclidean distances between items after their mapping, by $\phi$, to the feature space. Since for most usual kernels the feature space has infinite dimension, the computation of the distance cannot be directly performed in the feature space. Fortunately, the kernel trick allows to reduce this to a computation in input space. Indeed, if the kernel $K$ is positive definite then $K(\mathbf{x}_1, \mathbf{x}_2) =< \phi(\mathbf{x}_1)), \phi(\mathbf{x}_1)) >$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are two items in the input space. The distance between their images in feature space is $d(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)) = ||\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)|| = \sqrt{< (\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)), (\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)) >}$. After expansion and substitution using $K$, the following simple expression is obtained:

$$d(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)) = \\ = \sqrt{K(\mathbf{x}_1, \mathbf{x}_1) + K(\mathbf{x}_2, \mathbf{x}_2) - 2K(\mathbf{x}_1, \mathbf{x}_2)} \quad (1)$$

Now, consider the hyperplane $\mathbf{H}$ in feature space associated to the discrimination frontier of an SVM. It is defined by (see (Schölkopf and Smola, 2002))

$$\left| \sum_i \alpha_i \, y_i \, K(\mathbf{p}, \mathbf{x}_i) \; + \; b \right| = 0 \quad (2)$$

where $\mathbf{x}_i$ are the support vectors, $y_i$ the associated labels ($+1$ or $-1$), $\alpha_i$ the corresponding coefficients and $b$ the offset. Then, for some point $\mathbf{p}$ in the original space, the distance in feature space between its image $\phi(\mathbf{p})$ and the hyperplane $\mathbf{H}$, $d(\mathbf{p}, \mathbf{H})$, can be written as

$$d(\mathbf{p}, \mathbf{H}) \; = \; \frac{|\sum_i \alpha_i \, y_i \, K(\mathbf{p}, \mathbf{x}_i) \; + \; b|}{\sqrt{\sum_{i,j} \alpha_i \, \alpha_j \, y_i \, y_j \, K(\mathbf{x}_i, \mathbf{x}_j)}} \quad (3)$$

where the denominator is the norm of the vector defining the hyperplane in feature space and $\mathbf{x}_i$, $\mathbf{x}_j$ are support vectors.

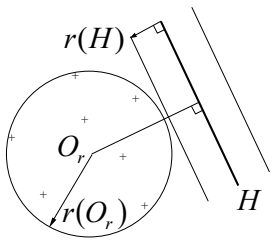


Figure 4: A sub-tree can be safely pruned if it does not intersect the query range, $d(O_r, H) > r(H) + r(O_r)$.

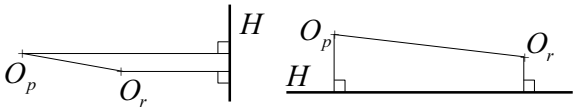


Figure 5: While $d(O_r, H) + d(O_r, O_p) \geq d(O_p, H)$ and $d(O_p, H) + d(O_r, O_p) \geq d(O_r, H)$ are always true, $d(O_r, H) + d(O_p, H) \geq d(O_r, O_p)$ does not necessarily hold.

Consider now an FQ of range $r(H)$, i.e. attempting to retrieve the items whose image in feature space is within a distance range of $r(H)$ to $\mathbf{H}$. If $\mathbf{p}$ corresponds to a routing node in the M-tree, then the sub-tree under the node of center $\mathbf{p}$ and radius $r$ can be further ignored (pruned) if $d(\mathbf{p}, \mathbf{H}) > r(H) + r$ (Fig. 4). With the same notations as for the M-tree, it is easy to show that:

1. If $d(O_r, H) > r(H) + r(O_r)$ then the sub-tree $T(O_r)$ can be safely pruned since, for each object $O_j$ in $T(O_r)$, $d(O_j, H) > r(H)$.

2. Unfortunately, $d(O_r, H) \geq |d(O_p, H) - d(O_r, O_p)|$ is false (Fig. 5) and only $d(O_r, H) \geq d(O_p, H) - d(O_r, O_p)$ holds, so it is actually necessary to compute $d(O_r, H)$ when $d(O_p, H) - d(O_r, O_p) \leq 0$ (i.e. in more cases than for classical point queries).

## 3.2 Search algorithms with hyperplane queries

The algorithms for answering exact $k$NN frontier queries can now be written. Since the FSM-tree is used to select, at every feedback round, *new* (unmarked) images, a mechanism is needed for excluding the images that were marked at previous rounds; it will be considered that the test predicate `isUnmarked`

only returns `true` for unmarked images. With the notations from the previous section and following the presentation of the original $k$NN search in (Zezula et al., 1996), (Ciaccia et al., 1997), the method for answering $k$NN frontier queries is given by Algorithms 1, 2 and 3.

**Algorithm 1** KNNFS ($T$: rootNode, $H$: query, $k$: nbNeighbors)

1: PR = [$T$, _ ]
2: NN[i] = random set of $k$ items
3: **while** PR not empty and NN[k].d $\geq d_{\min}$(head(PR)) **do**
4: NextNode = KNNChooseNode (PR)
5: KNNFSNodeSearch (NextNode, $H$, $k$)
6: **end while**

Here, `NN[k].d` is the distance field of the last entry in `NN` (i.e. the leaf entry having highest distance to the query among the entries already examined) and $d_{\min}$`(head(PR))` is the $d_{\min}$ field of the first entry in the `PR` list (i.e. the non-leaf entry having lowest $d_{\min}$ among the entries not yet examined).

**Algorithm 2** KNNSChooseNode (PR: priorityQueue): node

1: $d_{\min}(T(O_r^*)) = \min\{d_{\min}(T(O_r))\}$ over all entries in PR
2: remove entry $[T(O_r^*), d_{\min}(T(O_r^*))]$ from PR
3: return $T(O_r^*)$

To obtain the corresponding approximate $k$NN retrieval versions (see Section 2.3), the following changes should be performed:

- For AC-NN search, in Algorithm 1, line 3, `NN[k].d` is replaced with `NN[k].d`$/(1+\epsilon)$ and in Algorithm 3, lines 3 and 6, $d_k$ is replaced with $d_k/(1+\epsilon)$.

- For PAC-NN, in addition to the changes corresponding to AC-NN, search should be immediately stopped in Algorithm 3 if the $d_k$ value returned at line 16 satisfies $d_k \leq (1+\epsilon) r_{q,\delta}$.

Based on the same principles, we also developed similar algorithms for "maximum" queries. Our implementations rely on the M-tree package (Patella et al., 2000).

**Algorithm 3** KNNFSNodeSearch ($N$: node, $H$: query, $k$: nbNeighbors)

1: **if** $N$ is not a leaf **then**
2: **for all** $O_r \in N$ **do**
3: **if** $d(O_p, H) - d(O_r, O_p) \leq d_k + r(O_r)$ **then**
4: Compute $d(O_r, H)$
5: Compute $d_{\min}(T(O_r))$
6: **if** $d_{\min}(T(O_r)) \leq d_k$ **then**
7: Add $[T(O_r), d_{\min}(T(O_r))]$ to PR
8: **end if**
9: **end if**
10: **end for**
11: **else**
12: **for all** $O_j \in N$ **do**
13: **if** isUnmarked$(O_j) \wedge d(O_p, H) - d(O_j, O_p) \leq d_k$ **then**
14: Compute $d(O_j, H)$
15: **if** $d(O_j, H) \leq d_k$ **then**
16: $d_k$ = NNUpdate $([O_j, d(O_j, H)])$
17: **end if**
18: **end if**
19: **end for**
20: **end if**

# 4 Experimental evaluation

To evaluate the method put forward above, experimental comparisons were performed with exhaustive search on two ground-truth image databases, containing respectively 3,744 and 110,000 images. The use of a ground truth is required because the user is emulated during the RF sessions: the membership of images to the classes of the ground truth must be known by the emulated user if she is to provide reliable feedback.

The efficiency of our method was evaluated against exhaustive (sequential) search. The existing proposals dealing with the retrieval of the nearest points to a class frontier are (Panda et al., 2006) and (Panda and Chang, 2006); the very brief description of the clustering method in (Panda et al., 2006) does not enable reimplementation and the data structures suggested in (Panda and Chang, 2006) for indexing in the input space, such as the R*-tree, are known to be ineffective for high-dimensional input spaces (as the image description space we employ).

## 4.1 Experimental setup

To enable future comparisons, the databases employed are publicly available and have a non-

controversial ground truth. Since RF algorithms must contribute to a reduction of the semantic gap, it is necessary to avoid having too many "trivial" classes, for which simple low-level visual similarity is sufficient for correct classification. For an evaluation of RF it should nevertheless be possible to identify the target class relatively well with a limited number of positive and negative examples. With these criteria in mind, the two databases retained are:

- GT72 (3744 images), composed of the 52 most difficult classes—in terms of internal diversity within classes and of separability between classes—from the well-known Columbia color database (COIL 100), where each class contains 72 images.
- Amsterdam Library of Object Images (ALOI, 110,250 images, (Geusebroek et al., 2005)), a color image collection of 1,000 objects of various complexities where the viewing angle, illumination angle and illumination color for each object were systematically varied in order to produce about 110 images for each object. Similarity-based retrieval is more difficult for ALOI than for GT72 because there is more diversity within each class and there are many more classes.

The following generic descriptors are employed for the visual content of the images: a Laplacian weighted color histogram (Boujemaa et al., 2001), a probability weighted color histogram (Boujemaa et al., 2001), a classic HSV color histogram, a texture histogram (Ferecatu, 2005) relying on the Fourier transform and a shape feature (Ferecatu, 2005) inspired by the Hough transform. Weighted color histograms are a low-cost solution for taking into account local color uniformity. The texture histogram is based on the application of the Fourier transform to an image and describes the presence of different frequencies along various angles. To obtain the shape feature for a color image, the gray-level image is first computed, then the direction of the gradient is found for every pixel and a reference point is considered; for every pixel, the angle of the gradient and the length of the projection of the reference point along the tangent line going through the pixel position are counted in a joint histogram that is the shape feature. Previous evaluations (Ferecatu et al., 2004), (Ferecatu, 2005) have shown that the joint use of these descriptors helps avoiding the *numerical gap* for generalist image databases like the ones we employ here, i.e. sufficient information allowing to discriminate between images is provided by these descriptors. Linear PCA is applied to the descriptors in order to reduce the dimension of the joint descriptor from more than 600 to about 150.

The first kernel we consider is the Gaussian one (or Radial Basis Function, RBF in the following figures), $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\big(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2\big)$. This classical kernel, often employed by default, is highly sensitive to the scale parameter $\gamma$ (the inverse of the variance of the Gaussian). Preliminary experiments performed with exact retrieval in order to select the optimal $\gamma$ parameter for the RBF kernel produced $\gamma = 0.1$ for the GT72 database and $\gamma = 0.5$ for ALOI.

The angular kernel (ANG in the figures), $K(\mathbf{x}_i, \mathbf{x}_j) = -\|\mathbf{x}_i - \mathbf{x}_j\|$, introduced for SVM in (Schölkopf, 2000), is a conditionally positive definite kernel (Berg et al., 1984). As shown in (Schölkopf, 2000), the convergence of SVM remains guaranteed with this kernel and distance-based methods in feature space can still be applied. Since $K(\mathbf{x}, \mathbf{x}) = 0$, $\forall \mathbf{x} \in \mathcal{I}$, the angular kernel can be employed in our setting, as explained in the appendix.

The use of the Laplace kernel (LAPL in the figures), $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\big(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|\big)$, was advocated for histogram-based image descriptors (Chapelle et al., 1999). The scale parameter is $\gamma$ again and was fixed, following (Ferecatu et al., 2004), to 0.001; with such a small value for $\gamma$ the "angular" part of the Laplace kernel entirely covers the domain of the input data, so this kernel is expected to behave very similarly to the angular kernel.

In (Ferecatu et al., 2004), the angular and Laplace kernels were shown to provide consistently better results than the RBF kernel for image retrieval with relevance feedback. It was nevertheless decided to compare them again here in order to study the relation between kernel properties and the speedup obtained. Note that for the three kernels the feature space has infinite dimension. The L1 norm was used in all cases.

An FSM-tree was built for every database and kernel, then all the experiments were performed. Following (Ciaccia et al., 1997), (Ciaccia and Patella, 2000), the parameters employed for the construction of the M-tree in feature space are: (i) number of candidates for sampling: 10% of the number of items (images); (ii) minimum node utilization: 0.4; (iii) promotion method: confirmed; (iv) root promotion method: minimal radius; (v) confirmed promotion method: minimal maximal radius (mM_RAD); (vi) mM_RAD promotion method: average; (vii) split method: hy-

perplane. On a standard PC running Linux, the construction of an FSM-tree takes about 10 seconds for the GT72 database and 2 minutes for the large ALOI database.

At every feedback round the selector must return $s = 9$ images that the emulated user should mark as "relevant" or "irrelevant". To implement the MAO selection criterion, for both databases the system must first return the $k = 20$ most ambiguous images (nearest to the query hyperplane) and then choose among them the $s = 9$ least redundant ones. Every search session is initialized by considering one "relevant" image (belonging to the target class) and $s - 1$ randomly selected "irrelevant" images. The final goal of every search session is to rank the "relevant" images before the "irrelevant" ones. To evaluate the speed of improvement of this ranking, a precision measure is computed as follows: let $n$ be the number of images in the target class; at every RF round, count the number of images from the target class that are found in the $n$ images considered as most positive by the current decision function of the SVM; this number is then divided by $n$. The "mean precision" reported is obtained by averaging the precision measure defined above over all the RF sessions.

## 4.2 Evaluation results

For both databases, the exact retrieval with the FSM-tree for $k$NN hyperplane queries produced an insignificant speedup with respect to the evaluation of all the items in a database. This result was expected, given the challenging conditions: relatively concentrated distribution of distances (implying low selectivity for the FSM-tree), with hyperplane queries also having a lower selectivity than point queries.

The approximate retrieval methods, AC-$k$NN and PAC-$k$NN, were evaluated next. If the returned images are only approximate $k$NN of the query hyperplane, then these images are suboptimal in conveying information regarding the target class from the user to the system, so a loss in precision can be expected. The gain in retrieval speed during the selection stage of every feedback round was measured using the ratio of distance computations with the FSM-tree to distance computations with exhaustive search: the lower the ratio, the higher the gain in retrieval speed. The corresponding loss in precision was evaluated by depicting, for each kernel, both the evolution of the mean precision with the exact $k$NN retrieval and with PAC-$k$NN retrieval. Every couple of values for the $\epsilon$ and $\delta$ parameters define a specific trade-off between the gain in retrieval speed and the loss in precision.

With AC-$k$NN, a significant speedup was only obtained for high values of $\epsilon$, with a strong negative impact on the precision measure defined above. There was no valuable trade-off between speedup and precision loss, so these results are not reported. The values of the $\epsilon$ and $\delta$ parameters defining the PAC-$k$NN approximation were then explored. The gain in retrieval speed with $\epsilon = 0.1$ and $\delta = 0.15$ is significant, as shown in Fig. 6 for GT72 and Fig. 7 for ALOI. Since both databases hold into main memory, the speedup evaluated only concerns the number of distance computations and directly translates into gains in computation time. Notably, with PAC-$k$NN the system provided answers in real-time even on the large ALOI database.

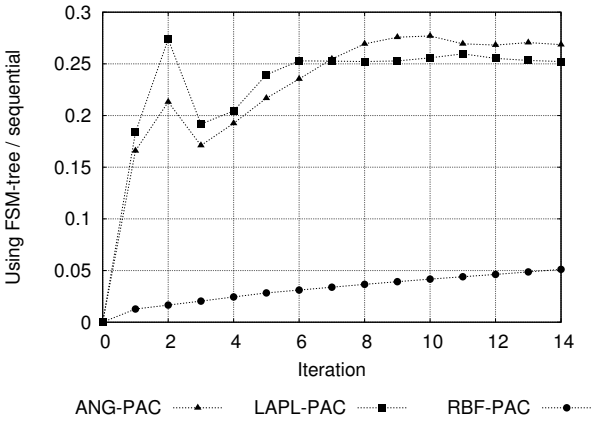


Figure 6: For the GT72 database and the 3 kernels (ANGular, LAPLace and RBF), the ratio of distance computations with the FSM-tree and PAC-$k$NN to distance computations with full evaluation (sequential scan) as a function of feedback rounds.

The loss in precision for these values of $\epsilon$ and $\delta$ is limited, as shown in Fig. 8 for GT72 and in Fig. 9 for ALOI, so the trade-off between speedup and loss of precision can be considered interesting.

The Laplace and angular kernels behave similarly, as expected for $\gamma = 0.001$. Note that for the angular kernel $d(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) = \sqrt{\|\mathbf{x}_i - \mathbf{x}_j\|}$, which links the distribution of distances computed in the feature space to the distribution of distances in the input space.

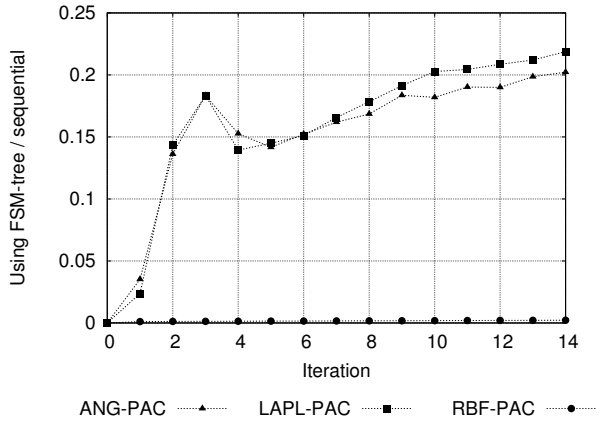


Figure 7: For the ALOI database and the 3 kernels (ANGular, LAPLace and RBF), the ratio of distance computations with the FSM-tree and PAC-$k$NN to distance computations with full evaluation (sequential scan) as a function of feedback rounds.

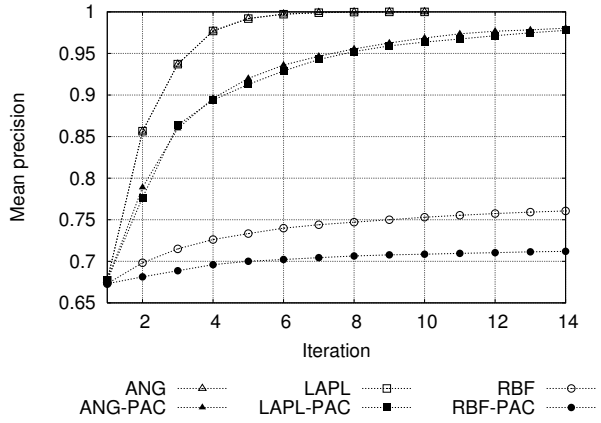


Figure 8: For the GT72 database, evolution of the mean precision obtained with exact search (empty marks) and PAC search (filled marks).

# 5 Discussion

The RBF kernel has a specific behavior, as seen from Fig. 6 to Fig. 9: the speedup is much higher than with the other two kernels, but the precision remains low and hardly improves after more feedback rounds. The RBF kernel has a fast decrease, so it is significantly different from 0 only at a distance lower than about three times its standard deviation $\frac{1}{\sqrt{\gamma}}$; to make class discrimination possible, the standard deviation must remain small. Therefore, the resulting SVM decision function is different from $b$ only in the vicinity of the

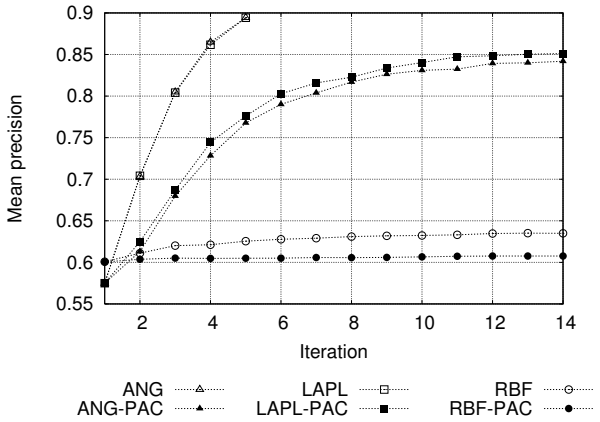


Figure 9: For the ALOI database, evolution of the mean precision obtained with exact search (empty marks) and PAC search (filled marks).

positive or negative examples, as shown in Fig. 10. For small $b$, approximate $k$NN retrieval quickly finds $k$ neighbors of the hyperplane query, but these selected items can come from almost anywhere since the decision function has a very small value for most of the unexplored input space. The selection is thus almost random and does not allow to improve precision with more feedback rounds.

Figure 10: For very local kernels, the decision function of the SVM can be close to 0 (light color in this picture) for a large part of the input space, so very many items can be mapped to the vicinity of the discrimination hyperplane in feature space.

The contribution of the approximation (PAC-$k$NN) is significant in speeding up retrieval, while maintain-

ing the quality of retrieval, in the very challenging conditions found here. As shown in Fig. 11, even if many spheres intersect the lower bound for the dynamic search range $r_{q,\delta}$, the improvement expected by pursuing the search under this bound can be very limited.

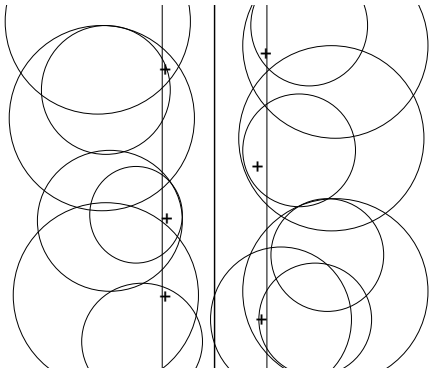

Figure 11: Even if many spheres intersect the query range, the improvement with respect to the approximate $k$ nearest neighbors can be very limited. This is the typical case when the selectivity of the FSM-tree is low.

# 6 Conclusion

We address the scalability of active learning based on 2-class SVM and applied to content-based image retrieval with relevance feedback. We put forward a search method that consists in performing approximate $k$NN *hyperplane* queries with an FSM-tree built in the feature space associated to the kernel of the SVM. The index and the query processing algorithms being defined in the feature space, the method can be directly applied to various types of data as long as an appropriate positive definite or conditionally positive definite kernel exists for the corresponding input space.

The evaluations performed show that search is significantly faster with this index, allowing real-time selection of the items returned to the user from a database of 110,250 images. These results also point out that approximate search can behave well in the challenging conditions of high (or infinite) dimensional spaces and less selective hyperplane queries. This method for the fast identification of ambiguous unlabeled examples is not limited to the RF context but can be employed for active learning in general. We are currently studying whether the low redundancy condition could be used during the search process rather than *a posteriori*. Also, it is important to identify kernels that can provide a better trade-off between the speedup obtained and the loss in precision.

# Acknowledgments

This work started while M. Crucianu and V. Oria were with the IMEDIA team at INRIA Rocquencourt. We wish to thank Nozha Boujemaa for providing the image descriptors developed by IMEDIA, and the authors of the M-tree library for making it publicly available. We are also grateful to the anonymous reviewers for their suggestions on the first version of the manuscript.

# References

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923.

Berg, C., Christensen, J. P. R., and Ressel, P. (1984). *Harmonic Analysis on Semigroups*. Springer-Verlag.

Boughorbel, S., Tarel, J.-P., and Boujemaa, N. (2005). Conditionally positive definite kernels for SVM based image recognition. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME'05)*, Amsterdam, The Netherlands.

Boujemaa, N., Fauqueur, J., Ferecatu, M., Fleuret, F., Gouet, V., Saux, B. L., and Sahbi, H. (2001). IKONA: Interactive generic and specific image retrieval. In *Proceedings of the International workshop on Multimedia Content-Based Indexing and Retrieval (MMCBIR'2001)*, pages 25–29, Rocquencourt, France.

Chang, E. Y., Li, B., Wu, G., and Goh, K. (2003). Statistical learning for effective visual image retrieval. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'03)*, pages 609–612, Barcelona, Spain.

Chapelle, O., Haffner, P., and Vapnik, V. N. (1999). Support-vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064.

Ciaccia, P. and Patella, M. (2000). Pac nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces. In *Proc. 16th International Conference on Data Engineering (ICDE 2000)*, pages 244–255, San Diego, CA.

Ciaccia, P., Patella, M., and Zezula, P. (1997). M-tree: an efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd IEEE International Conference on Very Large Data Bases (VLDB'97)*, pages 426–435, Athens, Greece.

Crucianu, M., Ferecatu, M., and Boujemaa, N. (2004). Relevance feedback for image retrieval: a short survey. In *State of the Art in Audiovisual Content-Based Retrieval, Information Universal Access and Interaction, Including Datamodels and Languages*. DELOS Network of Excellence.

Ferecatu, M. (2005). *Image retrieval with active relevance feedback using both visual and keyword-based descriptors*. PhD thesis, Université de Versailles, France.

Ferecatu, M., Boujemaa, N., and Crucianu, M. (2008). Semantic interactive image retrieval combining visual and conceptual content description. *Multimedia Systems*, 13(5-6):309–322.

Ferecatu, M., Crucianu, M., and Boujemaa, N. (2004). Retrieval of difficult image classes using SVM-based relevance feedback. In *Proceedings of the 6th ACM SIGMM International Workshop on Multimedia Information Retrieval*, pages 23–30, New York, USA.

Geusebroek, J. M., Burghouts, G. J., and Smeulders, A. W. M. (2005). The Amsterdam library of object images. *Int. J. Comput. Vision*, 61(1):103–112.

Heisterkamp, D. R. and Peng, J. (2003). Kernel VA-files for relevance feedback retrieval. In *Proceedings of the first ACM international workshop on Multimedia databases*, pages 48–54, New Orleans, LA, USA. ACM Press.

Hjaltason, G. R. and Samet, H. (1995). Ranking in spatial databases. In *SSD '95: Proceedings of the 4th International Symposium on Advances in Spatial Databases*, pages 83–95, London, UK. Springer-Verlag.

Li, J., Allinson, N. M., Tao, D., and Li, X. (2006). Multitraining support vector machine for image retrieval. *IEEE Transactions on Image Processing*, 15(11):3597–3601.

Panda, N. and Chang, E. Y. (2005). Exploiting geometry for support vector machine indexing. In *SDM*.

Panda, N. and Chang, E. Y. (2006). Efficient top-k hyperplane query processing for multimedia information retrieval. In *Proceedings of the 14th ACM international conference on Multimedia*, pages 317–326, New York, NY, USA. ACM Press.

Panda, N., Goh, K.-S., and Chang, E. Y. (2006). Active learning in very large databases. *Multimedia Tools and Applications*, 31(3):249–267.

Patella, M., Ciaccia, P., and Zezula, P. (2000). M-tree library. http://www-db.deis.unibo.it/Mtree/.

Peng, J. and Heisterkamp, D. R. (2003). Kernel indexing for relevance feedback image retrieval. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'03)*, Barcelona, Spain.

Rijsbergen, C. J. v. (1979). *Information retrieval*. Butterworths, London, 2nd edition.

Samet, H. (2006). *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Schölkopf, B. (2000). The kernel trick for distances. In *Advances in Neural Information Processing Systems*, volume 12, pages 301–307. MIT Press.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J. C., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471.

Schölkopf, B. and Smola, A. (2002). *Learning with Kernels*. MIT Press.

Tao, D., Li, X., and Maybank, S. J. (2007). Negative samples analysis in relevance feedback. *IEEE Trans. on Knowl. and Data Eng.*, 19(4):568–580.

Tao, D., Tang, X., and Li, X. (2008). Which components are important for interactive image searching? *IEEE Trans. on Circuits and Systems for Video Technology*, 18(1):3–11.

Tao, D., Tang, X., Li, X., and Wu, X. (2006). Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(7):1088–1099.

Tax, D. M. J. and Duin, R. P. W. (1999). Support vector domain description. *Pattern Recognition Letters*, 20(11-13):1191–1199.

Tong, S. and Chang, E. (2001). Support vector machine active learning for image retrieval. In *Proceedings of the 9th ACM International Conference on Multimedia*, pages 107–118, Ottawa, Canada. ACM Press.

Tong, S. and Koller, D. (2000). Support vector machine active learning with applications to text classification. In *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 999–1006, Stanford, CA, US. Morgan Kaufmann.

Zezula, P., Ciaccia, P., and Rabitti, F. (1996). M-tree: a dynamic index for similarity queries in multimedia databases. Technical report, CNUCE-CNR, Pisa, Italy.

Zhou, X. S. and Huang, T. S. (2001). Comparing discriminating transformations and SVM for learning during multimedia retrieval. In *Proceedings of the 9th ACM international conference on Multimedia*, pages 137–146, Ottawa, Canada. ACM Press.

# Appendix: Extension to a class of conditionally positive definite kernels

To extend the proposed approach from positive definite kernels to a class of conditionally positive definite (cpd) kernels, it is sufficient to show how to compute the distance in feature space between two items (1) and between an item and a hyperplane (3) for that class of cpd kernels.

All cpd kernels can be used with SVM, as (Schölkopf, 2000) shows. If $K$ is cpd and $K(\mathbf{x},\mathbf{x}) = 0$, $K(\mathbf{x},\mathbf{y}) \neq 0$, $\forall \mathbf{x},\mathbf{y} \in \mathcal{I}, \mathbf{x} \neq \mathbf{y}$, then the associated kernel $\tilde{K}(\mathbf{x}_1,\mathbf{x}_2) = K(\mathbf{x}_1,\mathbf{x}_2) - K(\mathbf{x}_1,\mathbf{x}_0) - K(\mathbf{x}_2,\mathbf{x}_0) + K(\mathbf{x}_0,\mathbf{x}_0)$ for some $\mathbf{x}_0 \in \mathcal{H}$ is positive definite, as shown in (Berg et al., 1984).

Let $\tilde{\alpha_i}$ and $\tilde{b}$ be the parameters estimated for the SVM when $\tilde{K}$ is substituted to $K$. As shown in (Boughorbel et al., 2005), $\tilde{\alpha_i} = \alpha_i$, i.e. the support vectors and the associated weights are the same. It can be shown that $\tilde{b} = b + \sum_i \alpha_i y_i K(x_i, x_0)$. These two properties prove that when using either $(K, \alpha_i, b)$ or $(\tilde{K}, \tilde{\alpha_i}, \tilde{b})$ in (2) the same decision function is obtained. Since the associated kernel $\tilde{K}$ is positive definite, it can be used as previously in (1) to derive the distance $d(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)) = \sqrt{\tilde{K}(\mathbf{x}_1,\mathbf{x}_1) + \tilde{K}(\mathbf{x}_2,\mathbf{x}_2) - 2\tilde{K}(\mathbf{x}_1,\mathbf{x}_2)}$. After substitution of $\tilde{K}$ and simplification, we obtain the simple expression $d(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)) = \sqrt{-2K(\mathbf{x}_1,\mathbf{x}_2)}$. Cpd kernels for which $K(\mathbf{x},\mathbf{x}) = 0$, $K(\mathbf{x},\mathbf{y}) \neq 0$, $\forall \mathbf{x},\mathbf{y} \in \mathcal{I}, \mathbf{x} \neq \mathbf{y}$ can thus be interpreted as being directly related to the Euclidean distance associated to the inner product $\tilde{K}(\mathbf{x},\mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$.

This also implies than $(\tilde{K}, \tilde{\alpha_i}, \tilde{b})$ and $(K, \alpha_i, b)$ are equivalent in defining the distance between an item and an hyperplane following (3).

To summarize, the two key distances that are sufficient for the proper definition of our approach can be computed equally well for those *conditionally* positive definite kernels for which $K(\mathbf{x},\mathbf{x}) = 0$, $K(\mathbf{x},\mathbf{y}) \neq 0$, $\forall \mathbf{x},\mathbf{y} \in \mathcal{I}, \mathbf{x} \neq \mathbf{y}$.